COMPARACIÓN DEL DESEMPEÑO DE LOS PROTOCOLOS DE ENRUTAMIENTO AODV Y DSR SOBRE UNA RED MANET EXPERIMENTAL

AUXILIAR DE INVESTIGACIÓN:

ESTEFANÍA ARAGÓN MONROY

CÓDIGO 1400458

DIRECTOR:

ING. ÁNGELA MARCELA MEJÍA FAJARDO PH.D

UNIVERSIDAD MILITAR NUEVA GRANADA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES

BOGOTÁ

2012

MARCO TEÓRICO

Las redes Ad Hoc son redes inalámbricas que se forman espontáneamente, se conforman de terminales inalámbricos móviles, que se comunican mediante enlaces inalámbricos y no poseen infraestructura fija, por lo que no hay un control centralizado pero gran flexibilidad. [13],[20]

Las redes Ad Hoc se pueden clasificar en redes de sensores, redes mesh, redes vehiculares (VANET, vehicular ad hoc network) y redes móviles Ad Hoc (MANET). Nuestro proyecto se centrará en las redes MANETs. [13]

En las redes MANETs la topología cambia constantemente, ya que los nodos móviles pueden dejar la red o ingresar a ella en cualquier momento, a la hora de necesitar hacer la conexión entre un nodo A y un nodo B se verifica que se encuentran dentro del rango de cobertura de cada uno, en dado caso si los nodos que se desean comunicar están muy lejos pueden usar múltiples saltos por nodos intermedios que colaboran para hacerlo, por lo que los nodos pueden servir de transmisores, enrutadores y receptores. [18], [19]

Para que los nodos puedan tener información de las rutas activas dentro de la topología de la red se usan protocolos de enrutamiento. Las características principales que deben poseer los protocolos de enrutamiento en redes MANETs son: implementación distribuída (ya que estas redes no deben depender de autoridades centralizadas), uso del ancho de banda (el protocolo no debe generar exceso de mensajes de enrutamiento), optimización de métricas (uso de balance de cargas, mínimo overead y adaptabilidad a la topología cambiante) y convergencia de ruta rápida (establecimiento de nuevas rutas estables al cambiar la topología de la red). [8]

Los protocolos de enrutamiento se dividen en tres grandes grupos, protocolos reactivos, proactivos e híbridos, nosotros nos vamos a centrar en los protocolos de enrutamiento reactivos AODV (Ad hoc On Demand Distance Vector) y DSR (Dynamic Source Routing), por ser los que mejores resultados presentan en diferentes condiciones ya que se han presentado como RFC (Request for Coments). El protocolo de enrutamiento DSR fue diseñado específicamente para ser usado en MANETs y hace que la red sea auto organizable y auto configurable, además, usa dos mecanismos para descubrir y mantener las rutas desde la fuente hasta el destino: Route discovery y Route maintenance [13]. El protocolo de enrutamiento AODV fue diseñado para redes Ad Hoc, AODV solo mantiene información de la ruta entre los nodos que necesitan comunicarse, pero no de

toda la ruta, se basa en la información que tienen los nodos sobre sus vecinos activos, los mecanismos que usa para descubrir y mantener las rutas son: Route request, Route reply, Route error y Route reply acknowledgment

Protocolo de enrutamiento AODV[22]

El protocolo AODV (Ad hoc On demand Distance Vector) es un protocolo de enrutamiento reactivo para redes Ad-Hoc, establece las rutas bajo demanda o cuando se necesita establecer comunicación entre dos nodos, y sólo mantiene información de la ruta que los comunica, es un protocolo de vector distancia, es decir, su métrica está basada en el número de saltos. Este protocolo es simple y liviano, busca las rutas cuando existe una solicitud, y escoge la ruta que responda primero, ya que asume que la respuesta se transmitió por la ruta más corta. Además, los mensajes de enrutamiento sólo tienen información sobre el Origen y el Destino, y no de la ruta completa, por lo que tienen un tamaño fijo, estos mensajes usan un número de secuencia de destino con el que se puede verificar si la ruta es actúal, también se usa para evitar bucles en la red.

Es importante mencionar que no siempre el mensaje que llega primero fue el que usó la ruta más corta, por lo que para reducir el ancho de banda utilizado, las rutas se mantienen activas hasta que no se necesiten, o falle algún enlace.

Descubrimiento de rutas

Route Request (RREQ)

Cuando el Nodo Origen desea enviar un paquete, lo primero que hace es revisar en su tabla de enrutamiento si tiene almacenada información sobre el siguiente salto que debe hacer para alcanzar el Destino, si no posee dicha información porque la ruta expiró o porque es la primera vez que desea enviar un paquete a este Destino y no conoce la ruta, inicia un proceso de descubrimiento de ruta.

En el proceso de descubrimiento de ruta, el Nodo Origen crea un mensaje llamado Route Request (RREQ), este mensaje contiene información del Nodo Origen, del Nodo Destino, un número de secuencia del Origen y un contador de saltos, el RREQ se difunde mediante un broadcast a sus

vecinos, estos a sus vecinos y así sucesivamente hasta llegar al Nodo Destino o hasta un nodo intermedio que tenga almacenada en su tabla de enrutamiento la ruta hasta el Destino.

Cuando un nodo recibe un RREQ almacena en su tabla de enrutamiento la dirección del vecino que se lo envió, con el fin de establecer la ruta inversa en un futuro. Si el nodo recibe un RREQ con el mismo número de secuencia de un RREQ que llegó con anterioridad lo desecha.

Formato de mensaje Route Request (RREQ)

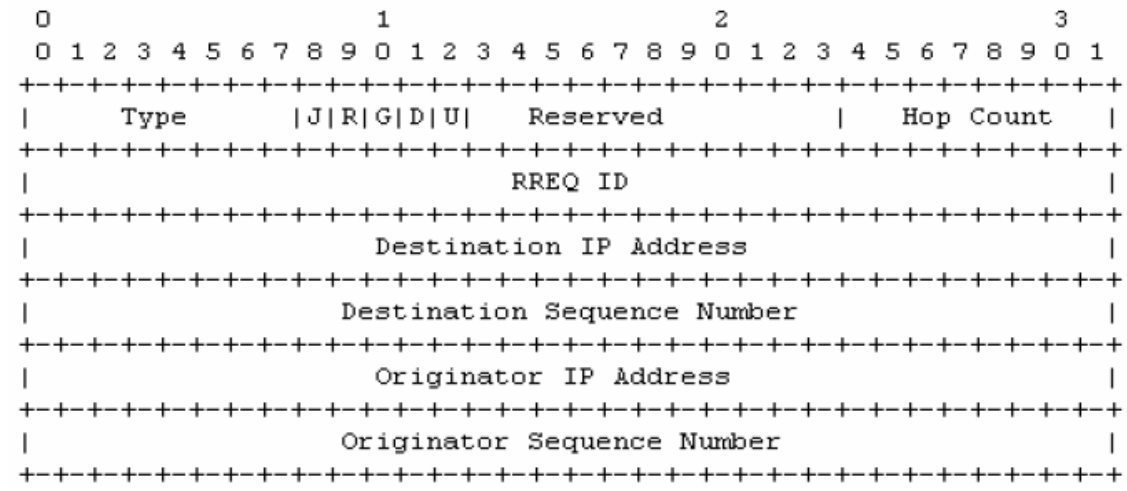


Fig. 1. En esta figura se muestra el formato del mensaje Route Request en donde el campo mutable es Hop Count.

Route Reply (RREP)

Cuando el RREQ llega al Nodo Destino este envía un mensaje ROUTE REPLY (RREP) al nodo que le envió primero el RREQ, y este retransmite de la misma manera el RREP, hacia atrás hasta llegar al Nodo Origen. En el caso que un nodo intermedio tenga almacenada la ruta hasta el Nodo Destino, este nodo se encarga de transmitir el RREP hacia el Nodo Origen. Si el Nodo Origen recibe otra ruta igual de actual como la que tiene ya almacenada, este nodo escoge la más corta y la actualiza para ser usada. Si por el contrario el Nodo Origen no recibe ningún RREP, este reenvía el RREQ aumentando el número de secuencia para que no lo desechen los otros nodos.

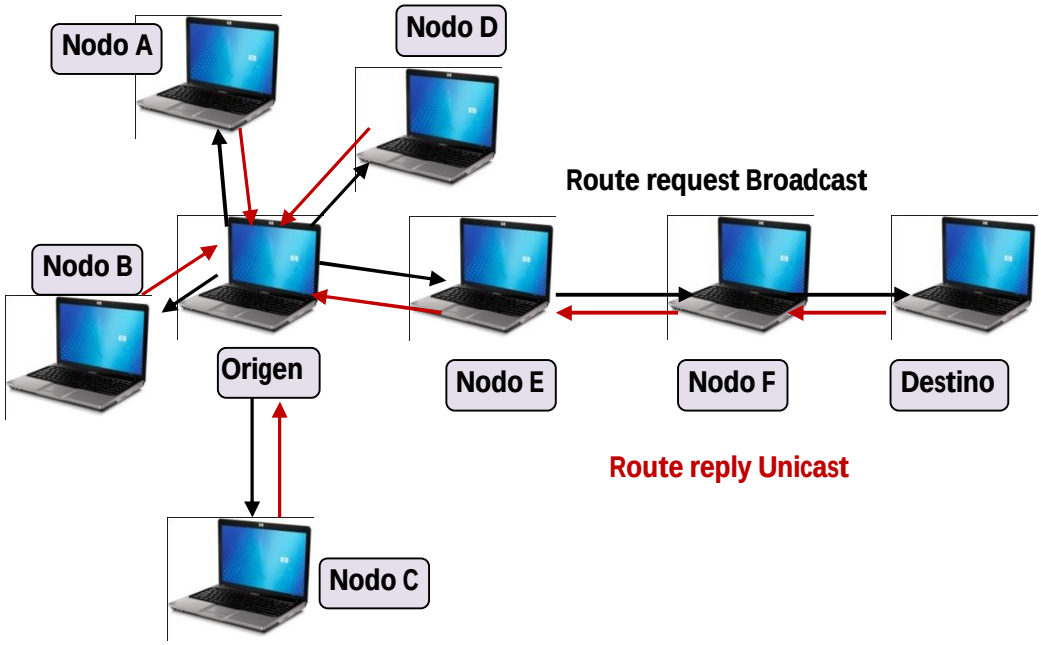


Fig. 2. Muestra el funcionamiento de los mensajes Route Request y Route Reply

Formato de mensaje Route Reply (RREP)

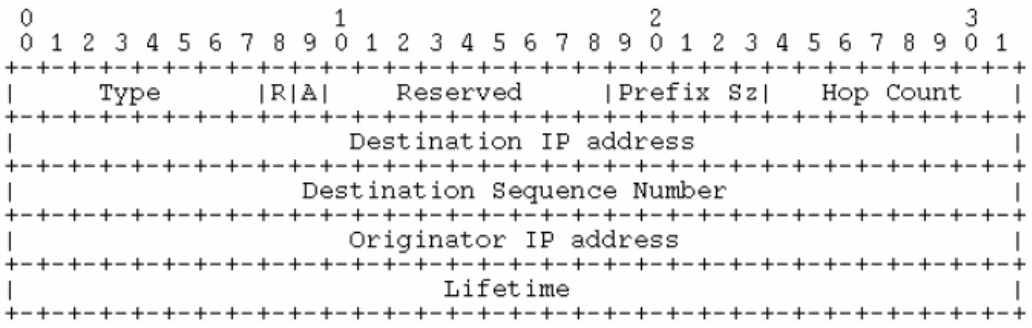


Fig. 3. Muestra el formato del mensaje Route Reply que posee el campo mutable Hop Count.

Formato de Mensaje Route Reply Acknowledgment (RREP-ACK)

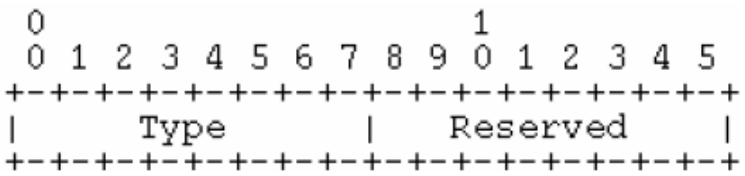


Fig. 4. Presenta el formato del mensaje Route Reply Acknowledgment que no posee campos mutables.

Los nodos almacenan información parcial de las rutas conocidas, lo que disminuye la memoria necesaria para funcionar. Los nodos mantienen una tabla de enrutamiento con los destinos conocidos, inicialmente está conformada de nodos vecinos y se le añaden nuevos destinos cuando un nodo necesita comunicarse con otro nodo que no está en su tabla, el nodo almacena en su tabla la ruta completa gracias al proceso de descubrimiento de ruta que inicia hacia el destino concreto, pero las tablas de enrutamiento tienen un tiempo de vida muy corto (*Lifetime)*.

Mantenimiento de rutas

El mantenimiento se realiza con el fin de evitar el uso de rutas erradas hacia Nodos Destino que ya no hacen parte de la red o que simplemente cambiaron de ubicación dentro de esta. El mantenimiento consiste en que las rutas duran cierto tiempo en la tabla de enrutamiento, pero pasado este tiempo son borradas, sin importar que hayan rutas todavía activas y que puedan alcanzar otros nodos.

Cuando cambia una ruta hacia un Destino porque un nodo se mueve, este se encarga de generar las peticiones de rutas en el momento que necesite enviar un paquete, con lo que informa su posición y nuevas rutas, pero de no pasar esto antes que el Nodo Origen envíe un paquete por dicha ruta, el primer vecino en darse cuenta se encargara de enviar un mensaje Route Error (RERR) hacia el Origen. En el mensaje RERR se coloca el valor de infinito a la métrica para que los nodos desechen la ruta.

Formato de mensaje Route Error (RERR)

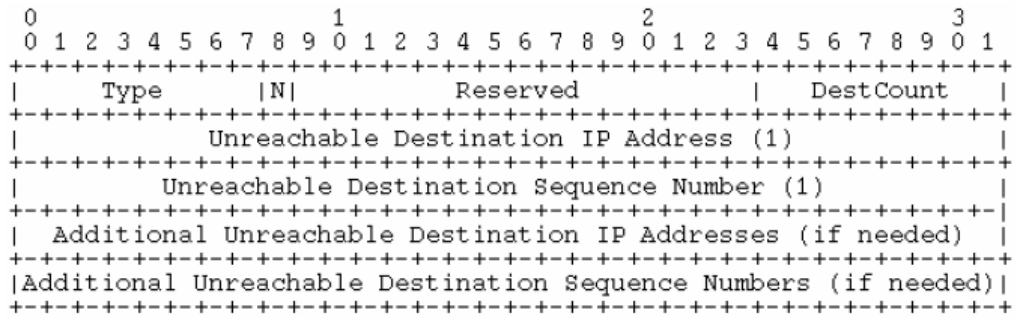


Fig. 5. Esta Imagen muestra el formato del mensaje Route Error que no presenta campos mutables.

AODV tiene la ventaja de reparar pequeñas fallas en la red antes de notificarlas a los demás, con el fin de evitar utilizar ancho de banda, por ejemplo si un vecino detecta por su cuenta que un

enlace se ha caído, tratara de determinar la ruta para llegar al nodo con el que perdió el enlace, si es capaz de encontrarla modifica la ruta de manera local sin notificar a nadie.

El protocolo hace uso de mensajes HELLO para realizar el mantenimiento de rutas, con este se determina la conectividad de los vecinos, pero el volumen de estos mensajes se debe reducir a solo los nodos que están transmitiendo datos, con el fin de no saturar la red, estos mensajes se envían periódicamente a través un broadcast.

Protocolo de Enrutamiento DSR[23]

El protocolo de enrutamiento DSR (Dynamic Source Routing) es un protocolo reactivo, bajo demanda, no almacena información sobre el estado de la red, únicamente rutas calculadas en cache a destinos que hacen parte de esta, es decir, que los nodos no se enteran del estado actual de la red, pero cuando se desea transmitir un mensaje, el nodo inicia una serie de procesos para definir la ruta hasta el destino, el nodo origen decide que ruta va a usar una vez analizadas las opciones, esta ruta se agrega al encabezado del paquete de datos y los nodos intermedios retransmiten el paquete al nodo siguiente, esto hace que se limiten los saltos pero los nodos intermedios disminuyen su procesamiento.

DSR posee dos mecanismos, uno para descubrir las rutas desde el nodo origen hasta el nodo destino, llamado Route Discovery, y otro para hacerle mantenimiento a las rutas ya encontradas con el fin de verificar que se encuentren activas en el momento de necesitarlas y además confirmar que los paquetes si llegan correctamente a su destino, llamado Route Maintenance.

Route Discovery

Este mecanismo se activa cuando un nodo origen necesita transmitir un paquete de datos hacia un nodo destino y no conoce la ruta. Inicialmente el nodo origen empieza a transmitir paquetes a sus vecinos denominados RREQ (Route Request), este paquete contiene el identificador del nodo origen, el del nodo destino y la ruta parcialmente calculada, este paquete inunda la red.

Cuando un RREQ llega a un nodo, este añade su identificador a la ruta contenida en el paquete, que es la ruta parcialmente calculada, y transmite a sus vecinos el nuevo RREQ, este proceso se lleva a cabo con todos los nodos de la red hasta que uno de los paquetes lo recibe el nodo destino. Cuando el nodo destino recibe el RREQ, transmite un Route Reply (RREP) hacia el nodo origen, usando la ruta inversa que tomo el RREQ, esto lo hace para informar al nodo origen de la ruta que

debe usar para transmitir los paquetes de datos. Una vez, el nodo origen recibe el RREP almacena en su cache la ruta que trae, y cuando finalmente va a transmitir un paquete con datos, le agrega un encabezado con la ruta que debe seguir hasta su destino.

Como se mencionó anteriormente, cada nodo almacena en su cache las rutas que conoce, esto se denomina Caching Overhead, lo que hace que cuando se desea transmitir un paquete, y el nodo origen conoce la ruta hacia el nodo destino, se reduzca el tiempo de entrega de los paquetes. Por lo anterior, en importante definir que rutas serán almacenadas en la cache:

Paquetes “escuchados”: La ruta se conoce y almacena porque un nodo escucha la retransmisión de un paquete desde el origen hasta el destino, sin intervenir en dicha retransmisión. Este mecanismo genera procesamiento adicional en los nodos intermedios pero se pueden descubrir mejores rutas y ayuda a reaccionar a fallas en enlaces.

Paquetes “enviados”: La ruta se conoce y almacena cuando un nodo retransmite un paquete.

Ruta almacenada durante el Route Request: La ruta se conoce y almacena cuando retransmite un RREQ, la ruta almacenada es la ruta parcial que va en el encabezado del RREQ.

Ruta almacenada en el Route Reply: Sucede lo mismo que en el caso anterior pero durante la propagación de RREP.

En el caso de que se transmita un RREQ y este llegue a un nodo intermedio que tiene almacenada en su cache la ruta hacia el nodo destino, este puede unir la ruta que tiene en su cache con la ruta parcial almacenada en el RREQ, verificando que no se repita ningún nodo en la nueva ruta generada, para cancelar el Route Request. Una vez el nodo intermedio cancele el Route Request inicia el Route Replay, a continuación se mostrará un ejemplo importante para explicar este proceso: Como se observa en la imagen el Route Request llega hasta el nodo C, ya que este nodo tiene almacenada en su cache la ruta hasta el Destino, este une las rutas como se mencionó anteriormente, verificando que no se repita ningún nodo, en este caso la ruta unida que se genera es: Origen-Nodo A-Nodo B-Nodo C-Nodo B-Nodo D-Destino, como se puede observar el Nodo B se repite formando un bucle inútil, por lo que la ruta se debe corregir, la Ruta corregida que se genera es: Origen-Nodo A-Nodo B-Nodo D-Destino, y finalmente esta ruta se envía en un Rute Replay hacia el Origen.

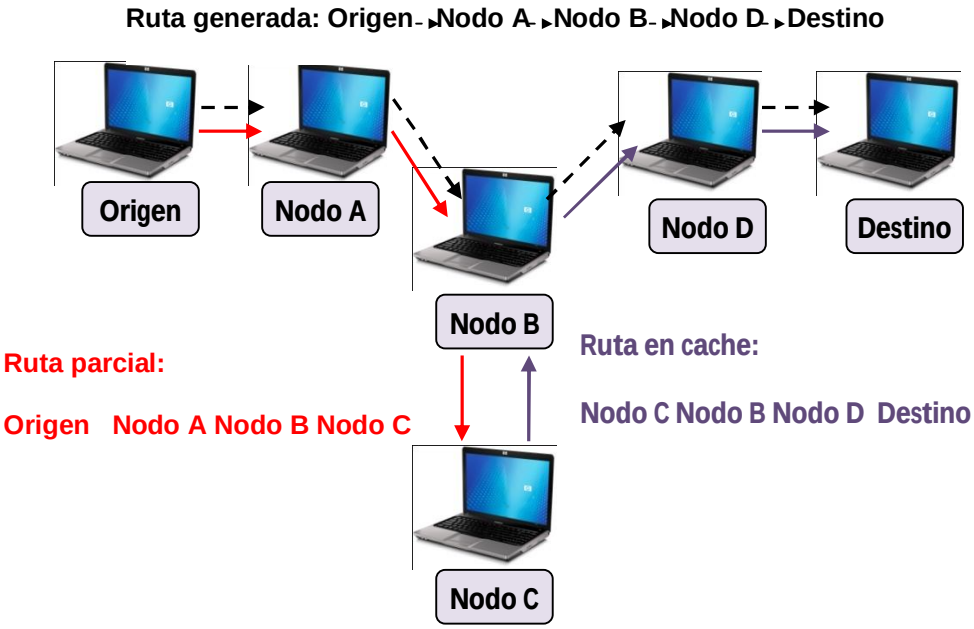


Fig. 6. Muestra el proceso de descubrimiento de rutas con el protocolo de enrutamiento DSR.

Por otro lado, también puede ocurrir que varios nodos tengan almacenados en su cache la ruta hasta el destino, y que estos respondan al tiempo con un Route Replay, lo que produce varias transmisiones al mismo tiempo, malgastando el ancho de banda y aumentando la probabilidad de colisión entre los paquetes. Para solucionar esto se da como opción que cada nodo espere un tiempo aleatorio en función del número de saltos hasta el destino, durante este tiempo el nodo Origen recogerá poco a poco las rutas, ordenadamente y reduciendo colisiones. Además, el nodo rechazará las rutas más largas.

Otra posible solución es limitar los saltos del Route Request, esto se hace incluyendo en el formato del paquete RREQ un campo que indique el número máximo de saltos que puede hacer, este valor decrementará en cada salto que se produzca durante el Route Request y si llega a cero se desecha el paquete.

Route Manteinance

Cuando se finaliza la etapa de descubrimiento de rutas, ósea cuando el Route Reply llega al origen con la ruta que se debe usar, el nodo origen inicia la transferencia de datos a través de esta. En

este punto inicia el proceso de mantenimiento de rutas, con el que se comprueba constantemente que los paquetes llegan a su destino exitosamente. Este proceso se explica mucho mejor con el siguiente ejemplo en donde en Nodo Origen inicia la transferencia de datos a través de la ruta escogida hacia el Nodo Destino.

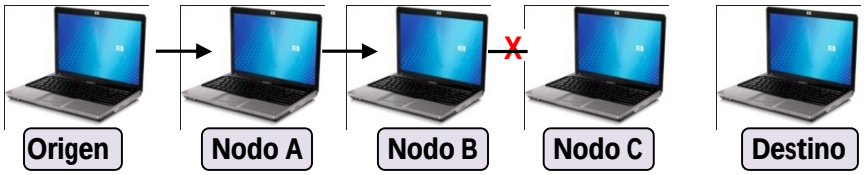


Fig. 7. Esta imagen muestra el proceso que se lleva a cabo cuando se le hace mantenimiento a las rutas que ya se habían descubierto.

Para realizar el mantenimiento de la ruta se hace un constante monitoreo de esta, verificando que el paquete que envía llega al siguiente nodo de forma exitosa, es decir el Nodo Origen comprueba que el Nodo A recibió el paquete, el Nodo A comprueba que el Nodo B recibió el paquete, y se hace el mismo proceso hasta que el paquete llegue al Nodo Destino.

Para comprobar que el paquete llegó exitosamente el nodo puede “escuchar” las retransmisiones, osea, si el Nodo Origen desea confirmar que el Nodo A recibió el paquete, el Nodo A puede escuchar si el Nodo B lo está retransmitiendo al Nodo C. Adicionalmente se suele incluir un bit de comprobación en el paquete, para que el siguiente nodo responda un ACK que confirme que ha recibido el paquete correctamente. Para este caso se debe tener en cuenta la simetría de la conexión, en otras palabras controlar si el enlace es unidireccional o bidireccional. Si el enlace es bidireccional no existe ningún problema, pero en caso de ser unidireccional el nodo tiene que mandar la comprobación por una ruta alternativa, lo que genera grandes inconvenientes.

Los nodos que están involucrados en la retransmisión de paquetes deben tener un sistema de control de envíos, esto se implementa controlando el número máximo de envíos por paquetes, si un nodo alcanza esta cifra considerará que el siguiente nodo está fallando, y enviara un Route Error hacia el nodo origen. Al ocurrir lo anterior los nodos actualizarán su cache para cambiar las rutas y que eviten ese enlace (identificado como roto). Una vez el Origen recibe el Route Error puede verificar si tiene una ruta alterna en la cache, ya que al enviar el Route Request le pudieron llegar varios Route Reply con diferentes rutas, en caso de no tener una ruta alterna se verá obligado a iniciar un nuevo Route Request para encontrar otra ruta e iniciar nuevamente el envío del paquete.

Los nodos pueden almacenar en la cache información sobre los enlaces que están rotos o también denominados nodos problemáticos, con el fin de no aceptar rutas que los contengan, con esto se garantiza que un Route Reply no contendrá información de enlaces que ya han generado errores, ya que pueden volver a generarlos.

Otra solución cuando se presenta un Route Error es que el nodo que no pudo entregar el paquete envíe el Route Error al nodo Origen, pero de inmediato intente "salvar" el paquete, esto lo hace buscando una ruta alterna, con lo que cambiaría la ruta original del paquete por la nueva ruta escogida de su cache. A continuación marca el paquete como "salvado", para que no se produzcan bucles y el paquete llegue al destino. Otra opción es que el paquete tenga como prefijo la ruta hasta donde se produjo el error y como sufijo se agregue la nueva ruta hasta el destino con lo que no se necesitaría marcar el paquete.

Con lo anterior, se logra optimizar la cache y se reduce el tiempo de envío de los paquetes, este tiempo también se puede reducir si se disminuye el número de retransmisiones de un paquete. Para lograr esto, durante el envío de los paquetes un nodo puede detectar una ruta más corta, ese nodo actualizará la ruta con el fin de reducir el número de retransmisiones y continúa la transmisión del paquete por la nueva ruta. Este proceso automático de reducción de rutas se lleva a cabo durante el Route Reply.

INSTALACIÓN Y CONFIGURACIÓN DEL PROTOCOLO DE ENRUTAMIENTO AODV

El protocolo de enrutamiento AODV se puso a prueba gracias a la implementación de la versión AODV-UU 0.9.6, actualmente es la última versión que existe del protocolo, desarrollada en el año 2011 por la Universidad de Uppsala, ubicada en la provincia de Uppsala en Suecia. AODV se desarrolló sobre una máquina virtualizada de Debian GNU/Linux 6.0.2 (squeeze) con kernel 2.6.32-5-686, a continuación se muestran los pasos para descargar y configurar AODV:

Con el siguiente comando se descarga el protocolo AODV, el protocolo viene en un paquete comprimido en el formato *.tar.gz*:

$wgethttp://sourceforge.net/projects/aodvuu/files/AODV-UU/0.9.6/aodv-uu-0.9.6.tar.gz

Una vez descargado el archivo se debe descomprimir, esto se realiza con el siguiente comando:

$tar –xzvf aodv-uu-0.9.6.tar.gz

Para verificar que archivo fue descomprimido se puede usar el comando:

$ ls

Y se podrán observar tanto el archivo comprimido como el archivo descomprimido:

```
aodv-uu-0.9.6  aodv-uu-0.9.6.tar.gz
```

Fig. 8. Muestra el archivo descomprimido y comprimido del protocolo de enrutamiento AODV.

A continuación se debe ingresar al archivo del protocolo:

$cd aodv-uu-0.9.6

Para observar el contenido del archivo aodv-uu-0.9.6, se puede usar el comando:

$ls

En la lista de contenido se puede acceder al archivo README, donde se pueden verificar las condiciones en que debe ser montado AODV y los pasos para la instalación y configuración del protocolo:

$ more README

```
Real world:
* Linux OS (2.4.x, 2.6.x).
* Kernel with Netfilter support.
  Most Red Hat/Fedora kernels have this support.
* Wireless LAN cards in ad-hoc mode (alternatively a wired setup can
  be used).
```

Fig. 9. Características que se deben verificar para la correcta instalación y configuración del protocolo de enrutamiento AODV.

Como se muestra en la anterior imagen el protocolo AODV debe ser montado sobre el sistema operativo Linux mientras tenga un kernel con versiones 2.4 o 2.6, tarjetas de red LAN en modo ad hoc y kernel con soporte netfilter en caso de implementar el protocolo directamente sobre el equipo físico.

Una vez verificado el archivo README, se ingresan los siguientes comandos para instalar los paquetes del protocolo AODV, antes se debe acceder como superusuario:

$su

make

makeinstall

Al ingresar los anteriores comandos la instalación finalizará con los siguientes comentarios, que indican que se ingresa al archivo del protocolo aodv para realizar la instalación de los paquetes de este, hecha la instalación sin surgir ningún error se finaliza el proceso saliéndose del archivo del protocolo, se debe aclarar que no deben aparecer errores.

Fig. 10. Muestran que el protocolo de enrutamiento se instaló sin presentar ningún error.

El siguiente comando se usa para cargar el módulo del protocolo AODV, llamado kaodv, en el kernel:

modprobe kaodv

Una vez cargado el módulo del protocolo, se deben tener en cuenta los siguientes datos para poner a correr el protocolo:

- l	Guarda los log´s con los que podemos ver todos los paquetes (RREQ, RERR, RREP, HELLO) que se envían los nodos.
-r	Guarda las tablas de enrutamiento de cada nodo. Se le debe especificar un valor (segundos), que indica la frecuencia con la que se actualiza la tabla de enrutamiento.
-i	Se utiliza para especificar la interfaz que debe utilizar el protocolo para ver los otros nodos.

Tabla 1. Parámetros que se pueden configurar para inicializar el protocolo de enrutamiento AODV.

En este caso la tabla de enrutamiento se actualizará cada 3 segundos y se comunicarán los nodos por la interfaz eth0:

aodvd –l –r 3 –i eth0

Los paquetes que envían los nodos se pueden encontrar en el archivo aodvd.log y todas las tablas de enrutamiento del nodo se pueden encontrar en el archivo aodvd.rtlog. Los archivos antes mencionados, generados por el protocolo, se pueden encontrar en: /var/log

Encontramos en el archivo aodv.log la siguiente información:

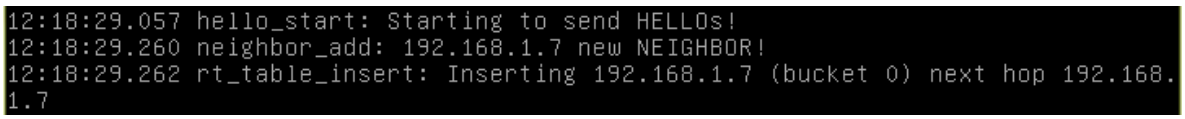


Fig. 11. Muestra el proceso para descubrimiento de vecinos a través de los mensajes HELLO.

En la anterior captura podemos observar que el protocolo de enrutamiento AODV inicia el proceso de descubrimiento de rutas enviando mensajes HELLOs en la red, y en seguida descubre un nuevo vecino con la dirección IP 192.168.1.7, y finalmente lo ingresa en la tabla como el siguiente salto.

Fig. 12. Muestra como se elimina vecinos de la tabla de enrutamiento del protocolo de enrutamiento AODV.

Con la anterior capturar podemos observar que se envía un mensaje HELLO al equipo con la dirección IP 192.168.1.7, y nunca se obtiene una respuesta por lo que se indica que falló el enlace y que el equipo no fue encontrado en consecuencia después de 15000 msegs el equipo es eliminado de la tabla de enrutamiento que ya se había construido.

Encontramos en el archivo aodv.rtlog la siguiente información:

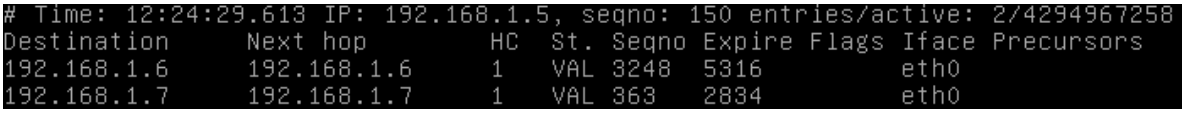


Fig. 13. Muestra la tabla de enrutamiento que arma el protocolo de enrutamiento AODV.

En la anterior captura, podemos observar la tabla de enrutamiento que ha construido el equipo con dirección IP 192.168.1.5, esta tabla posee un número de secuencia e indica el número de equipos que fueron registrados en la tabla (2), finalmente, muestra las características generales de dichos equipos como su dirección IP, número de secuencia, y la interfaz por la que se comunica con el equipo que está construyendo la tabla.

INSTALACIÓN Y CONFIGURACIÓN DEL PROTOCOLO DE ENRUTAMIENTO DSR

El protocolo de enrutamiento DSR se puso a prueba gracias a la implementación de la versión DSR-UU 0.2, actualmente, es la última versión que existe del protocolo, desarrollada en el año 2005 por la Universidad de Uppsala, ubicada en la provincia de Uppsala en Suecia. DSR se montó sobre Debian GNU/Linux Sarge con kernel 2.6.8-4-386, a continuación se muestran los pasos para descargar y configurar DSR:

Con el siguiente comando se descarga el protocolo DSR, el protocolo viene en un paquete comprimido:

$wgethttp://sourceforge.net/projects/dsruu/files/DSR-UU/0.2/dsr-uu-0.2.tar.gz

Una vez descargado el archivo se debe descomprimir, con el siguiente comando se descomprime el archivo:

$tar –xzvf dsr-uu-0.2.tar.gz

Para verificar que archivo fue descomprimido se puede usar el comando:

$ ls

El paso que sigue, es ingresar al archivo del protocolo, esto se realiza con el siguiente comando:

$cd dsr-uu-0.2

Al ingresar el protocolo se puede acceder al archivo README, donde se pueden verificar las condiciones en que debe ser montado DSR y los pasos para la instalación y configuración del protocolo:

$ more README

```
DSR-UU is a DSR implementation that runs in Linux 2.6 (2.4 support
obsolete) and in the ns-2 network simulator.
```

Fig. 14. Muestra las características que deben tener en cuenta para la instalación y configuración del protocolo de enrutamiento DSR,

Una vez verificado el archivo README, se ingresan los siguientes comandos para instalar los paquetes del protocolo DSR:

make

makeinstall

Para ingresar las instrucciones que corren el protocolo se debe acceder a la siguiente ruta:

cd /lib/modules/2.6.8-4-386/dsr

Se debe proseguir con los siguientes comandos, el primer comando se usa para la instalación del protocolo en el kernel y con el segundo comando se instalan los módulos que el protocolo usa:

insmodlinkcache.ko&&insmoddsr.koifname=eth0

Para que el protocolo pueda hacer transferencia de paquetes crea una interfaz llamada dsr0, esta interfaz se crea con el siguiente comando:

ifconfig dsr0 192.168.100.x up (donde X completa la dirección IP)

Se puede verificar la creación de la interfaz con el comando:

ifconfig

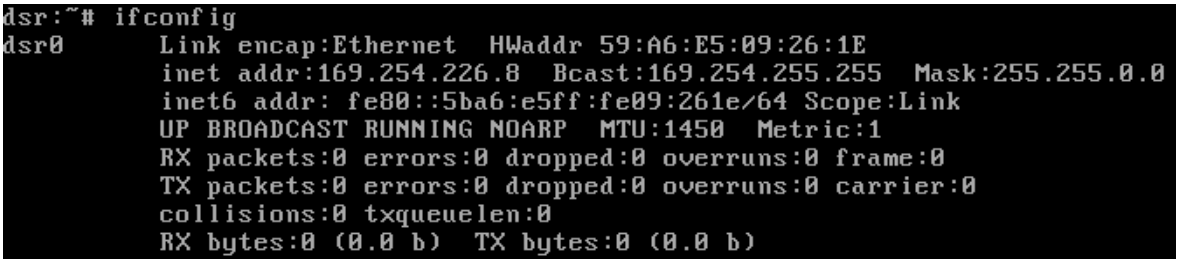


Fig.15. Muestra la interfaz que crea el protocolo de enrutamiento DSR.

Al ingresar este comando se genera un mensaje que confirma que el protocolo puede iniciar el intercambio de paquetes:

Fig. 16. Muestra el mensaje que se genera cuando la interfaz DSR0 se activa.

CONFIGURACIÓN ADICIONAL

Creación de la máquina virtual

Para iniciar, es necesario instalar un software de virtualización, en este caso se instaló el software Oracle VM VirtualBox desarrollado por la Corporación Oracle, además, se debe descargar el sistema operativo en donde se va a montar el protocolo de enrutamiento, el sistema operativo debe descargarse como un archivo comprimido con el formato .ISO. y proseguir con la configuración que se indica a continuación:

El primer paso, es ubicar en la ventana principal del software Oracle VM VirtualBox el icono 

, le damos click para crear una nueva máquina virtual, y seguimos los pasos que el asistente del software indica.

Durante el proceso se puede observar que el software tiene una gran variedad de sistemas operativos que puede crear y adicionalmente gran variedad de versiones de cada sistema operativo, gracias a esta variedad se pueden instalar y configurar los protocolos de enrutamiento sin importar que las versiones que usan del sistema operativo sean tan diferentes.

Fig. 17. Muestra la sección del software Oracle VM VirtualBox en donde se pueden escoger tanto el sistema operativo como la versión de este sistema operativa.

Una vez es creada la máquina virtual se observará la ventana principal del software como se muestra a continuación, en ella podemos tener acceso a las diferentes máquinas virtuales que se han creado incluídas las máquinas en donde se instalaron y configuraron los protocolos de enrutamiento:

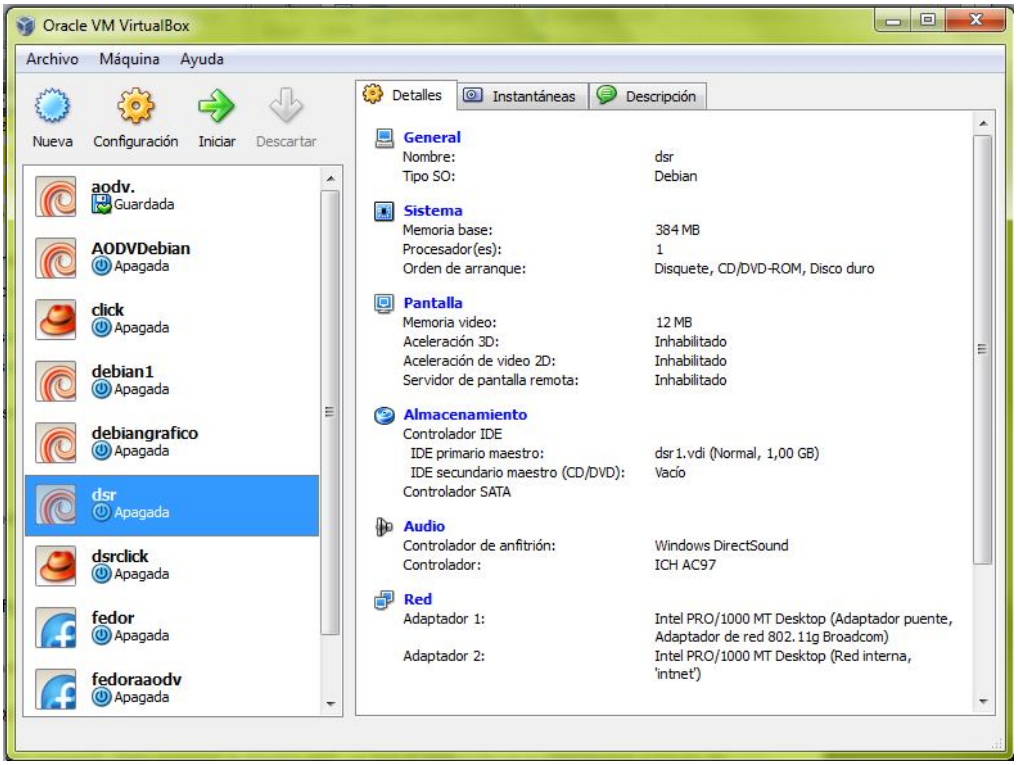


Fig. 18. Muestra la ventana principal del software Oracle VM VirtualBox en donde se pueden ver a mano izquierda las máquinas virtuales que se han creado.

Es importante verificar la configuración 

de la máquina virtual que se ha creado, esencialmente la red a la que se va a conectar. Al ingresar a la red se debe escoger en la opción de conectado a: Adaptador puente, y el Nombre corresponderá al nombre de la tarjeta de red que el equipo use, si no se realiza este proceso una vez se ponga a correr cualquier protocolo de enrutamiento no se podrán transmitir ni recibir mensajes de enrutamiento o de cualquier otro tipo.

En la siguiente captura se puede observar que la máquina virtual está conectada al equipo físico a través de un adaptador puente llamado Adaptador de red 802.g Broadcom.

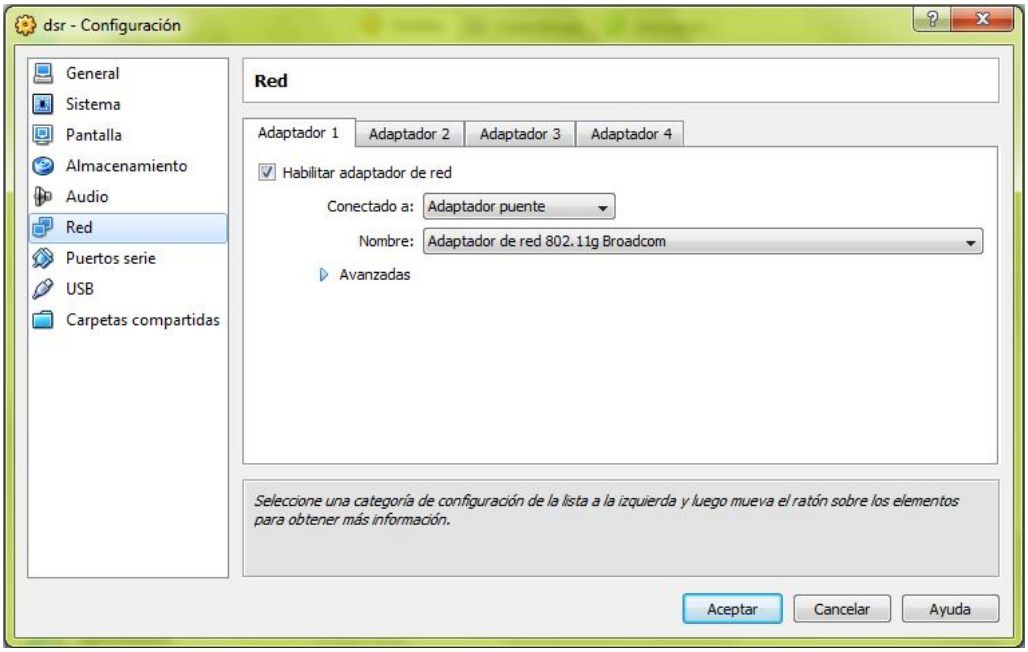


Fig. 19. Muestra la ventana que se despliega al elegir la opción Configuración.

Cuando se crea la máquina virtual se genera un disco de almacenamiento con el formato .vdi en donde se guardan todas las modificaciones y configuraciones que se realicen en la máquina virtual, este disco además, puede ser copiado en otro equipo portátil en caso de necesitar trasladar la máquina.

En el caso de necesitar pasar la máquina a otro equipo portátil se copia la máquina y se inicia el proceso para crear la máquina a partir de un disco duro existente como se muestra a continuación:

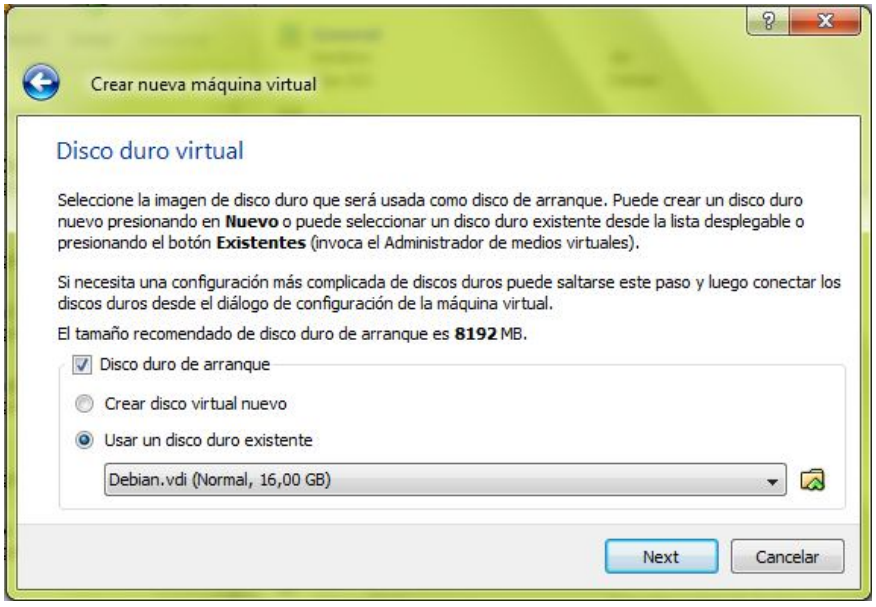


Fig. 20. Muestra la ventana en donde se debe elegir si se va a crear un disco duro virtual nuevo o se va a usar un disco duro existente.

Al iniciar 

la máquina virtual creada, sí se generan errores se debe verificar en la configuración 

que el disco duro usado se encuentre como un controlador IDE, como se muestra a continuación:

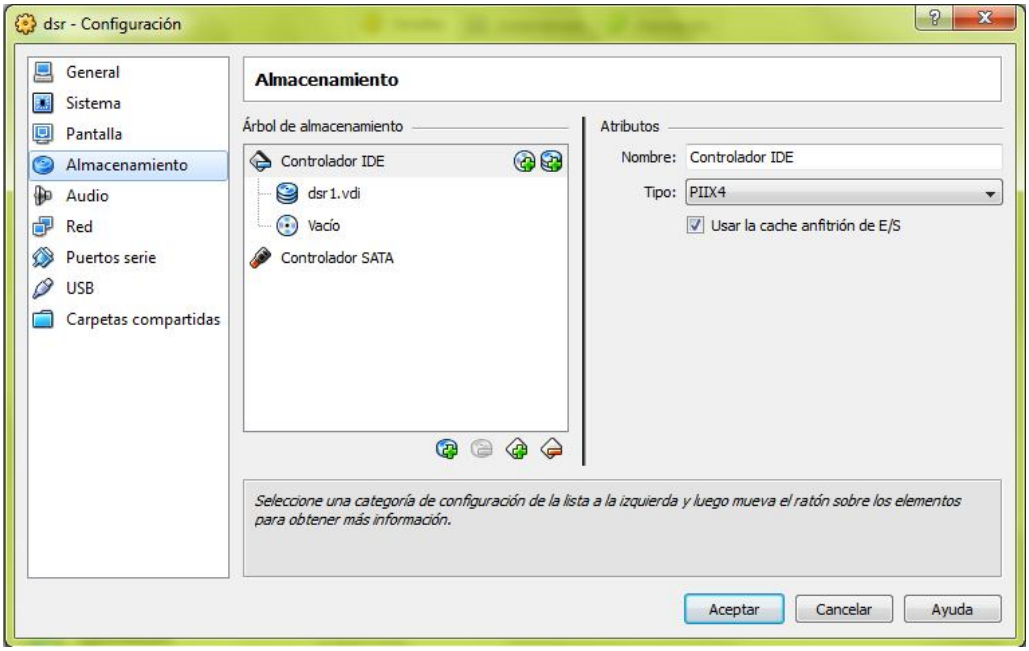


Fig. 21. Muestra la ventana del software Oracle VM VirtualBox en donde se puede observar y/o modificar el almacenamiento de la máquina virtual creada.

Se debe mencionar que gracias a que los protocolos de enrutamiento se instalan y configuran sobre una máquina virtual, se pueden ponen a correr sobre cualquier equipo físico a pesar de poseer un sistema operativo totalmente diferente. En este caso las máquinas virtuales con los protocolos se montaron sobre Windows 7 Ultimate, y para que finalmente se puedan probar sobre una red Ad Hoc se debe crear y configurar esta red en el equipo físico.

Crear una red Ad Hoc en Windows 7 Ultimate

Para crear una red Ad Hoc se debe ingresar al panel de control y escoger la opción Centro de Redes y Recursos Compartidos.

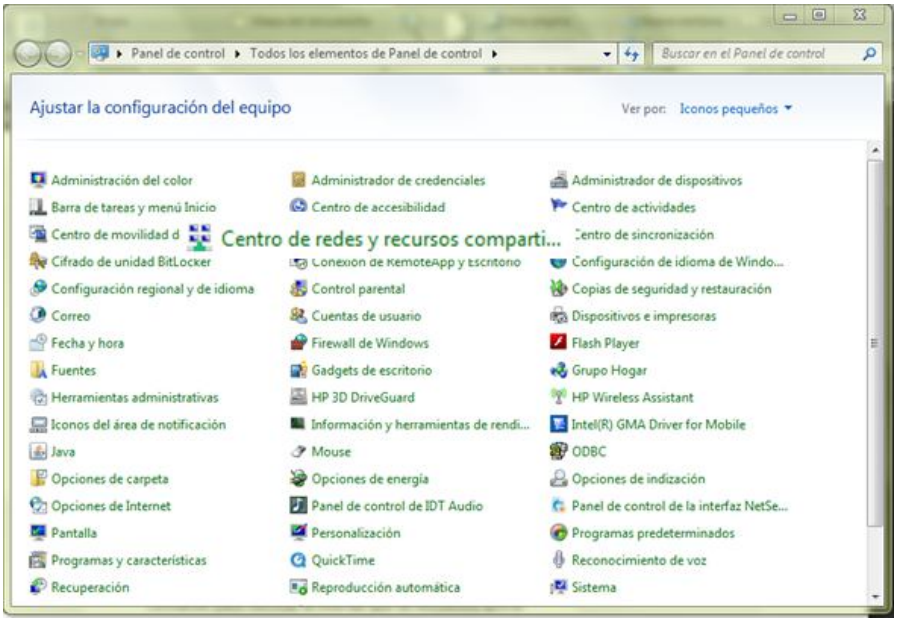


Fig. 22. Muestra la ubicación de la opción Centro de redes y recursos compartidos en el Panel de control.

Una vez ingresemos al centro de redes y recursos compartidos procedemos a acceder a la opción Administrar redes inalámbricas.

Fig. 23. Muestra la ubicación de la opción Administrar redes inalámbricas en el Centro de redes y recursos compartidos.

Al acceder al Administrador de redes inalámbricas aparecerá una lista con las redes que el equipo ha usado y en la parte superior de la ventana podemos observar la opción agregar, esta opción se usa para crear nuevas redes:

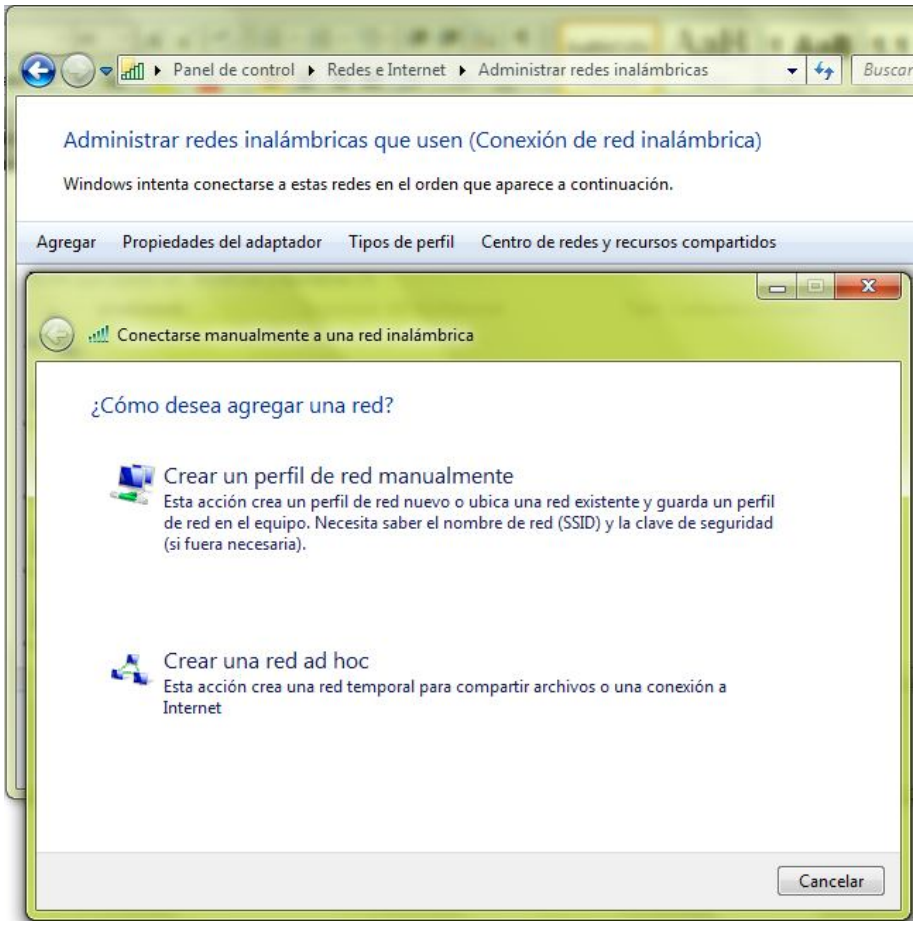
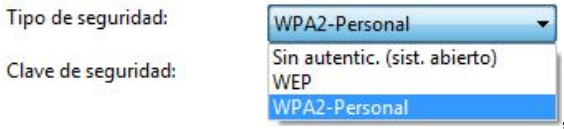


Fig. 24. Muestra la ventana en donde se encuentra la opción Crear una red ad hoc.

En este caso se escoge la opción crear una red Ad Hoc y se prosigue a nombrar la red y a darle la seguridad que se desee, los tipos de seguridad que se pueden escoger son



, se recomienda usar la seguridad WPA2-Personal, ya que está basado en un estándar de seguridad mas reciente (802.11i) en comparación con WEP que fue el primer estándar de seguridad para redes wifi y presenta una protección débil, hecho lo anterior se finaliza el proceso y el equipo quedará en espera de los usuarios que se deseen conectar a él.

Una vez creada la red Ad Hoc a la que se conectarán los equipos, para probar el comportamiento de los protocolos de enrutamiento, se debe ingresar a cada máquina virtual y verificar si la configuración actual de la interfaz, por la que se comunican al equipo físico, es correcta.

Configuración de la interfaz

Comando para verificar la interfaz que se encuentra activa:

#ifconfig

En este comando debe aparecer la interfaz eth0, ya que el protocolo va a comunicarse a través de esta interfaz, tal como se muestra a continuación:

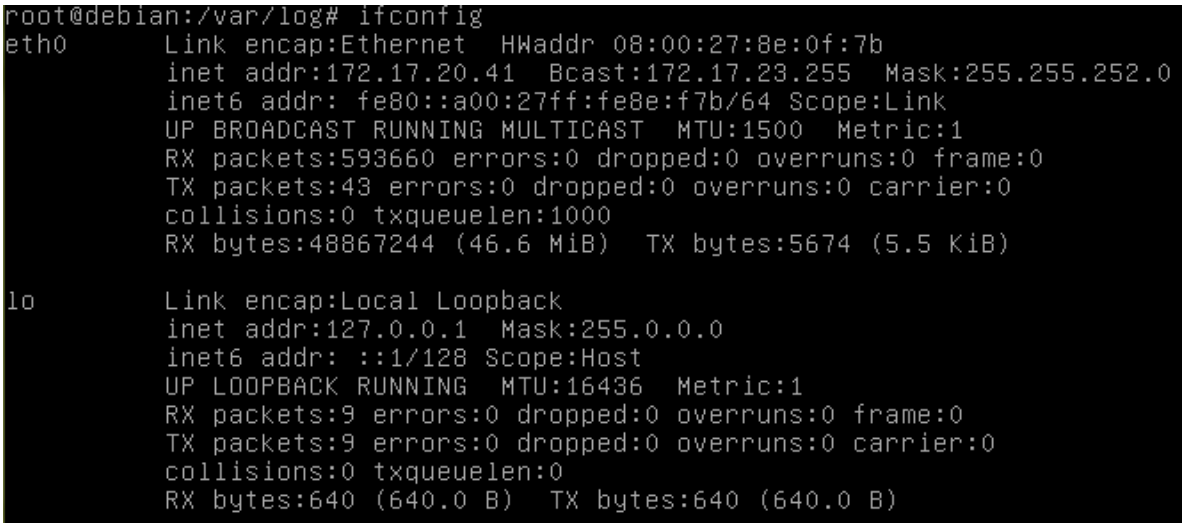


Fig. 25. Muestra la interfaz que se encuentra activa y que se podría configurar.

En caso de no poder observar la información completa se deben teclear al tiempo:

Shift + pg up (para desplazarse hacia arriba)

Shift + pgdn (para desplazarse hacia abajo)

En caso de que no aparezca la eth0 sino la eth1, se debe renombrar la interfaz, ya que en gran cantidad de programas se toma por defecto la interfaz 0 y si se corre dicho programa sin renombrar la interfaz se van a generar errores y no existirá comunicación con otros equipos, esto se hace de la siguiente manera:

Se debe ingresar a esta ruta /etc/udev/rules.d y con el siguiente comando se pueden realizar cambios como renombrar la interfaz:

#nano 70-persistent-net.rules

Una vez se ingresa se deben ubicar las siguientes líneas:

SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="XX:XX:XX:XX:XX:XX", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"

Y realizamos el cambio en el nombre de la interfaz remplazando eth1 por eth0, en caso de existir mas interfaces se deben comentariar con el fin de dejar activa únicamente la interfaz que se va a usar, en este caso la interfaz eth0, se inhabilitan las otras interfaces de la siguiente manera:

#SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="XX:XX:XX:XX:XX:XX",

#ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth2"

Para que los cambios se efectúen se deben guardar, salir del archivo, y reiniciar la máquina.

Una vez reinicie la máquina se puede comprobar si la interfaz que se encuentra activa es la eth0 con el comando:

#ifconfig

Si aún aparece la eth1, se debe realizar el proceso una vez más, y verificar nuevamente.

Ya que se activó la eth0, se puede proceder a configurarla, se le debe asignar una dirección IP, y "levantar" la interfaz:

#ifconfig eth0 192.168… (IP deseada) up

ANÁLISIS

ESCENARIO

Se decidió implementar este escenario para verificar que el protocolo de enrutamiento encuentra las rutas hacia el destino en particular, en este caso existían dos rutas posibles hacia él, una de las

opciones era usar el equipo con dirección IP 192.168.1.8 como nodo intermedio, y la otra era usar el equipo con dirección IP 192.168.1.6.

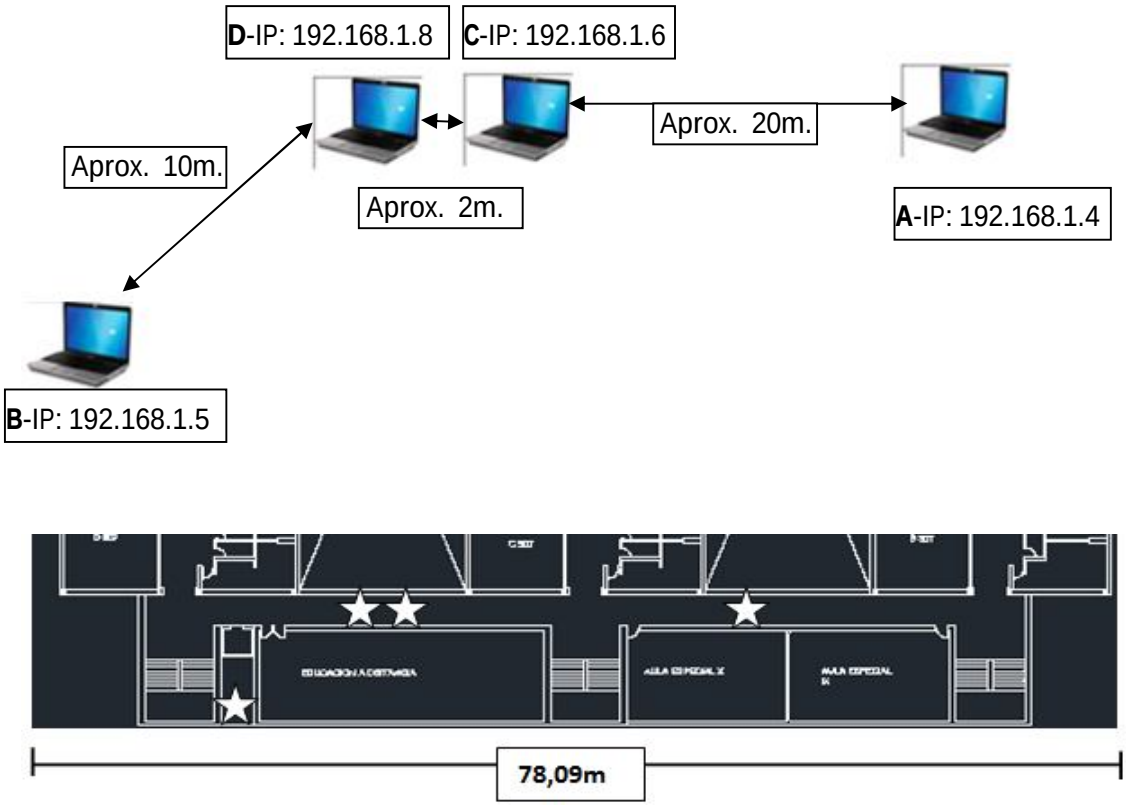


Fig. 26. Muestra el escenario que se montó para realizar las pruebas con los protocolos de enrutamiento.

Capturas en la red en el momento en el que los equipos se conectan a la red Ad Hoc.

Protocolos

LLMNR: Link Local Multicast Name Resolution, es un protocolo basado en DNS, resuelve los nombres de los sistemas informáticos cercanos, funciona en el entorno IPv4 o IPv6.[24]

En esta captura se puede observar que los equipos se conectan a la red Ad Hoc, y que se resuelve su nombre en la red, en este caso Usuario-PC.

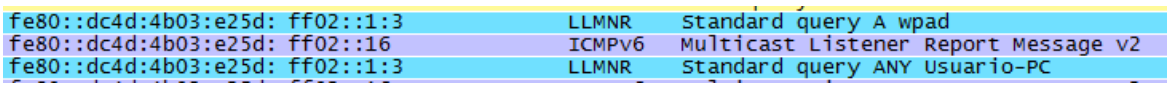


Fig. 27. Muestra una sección de la captura de wireshark en donde se resuelve en nombre del equipo llamado Usuario-PC.

ICMPv6: Internet Control Message Protocol, este protocolo tiene las siguientes características: Nuevo formato de encabezado, espacio de direccionamiento más grande, infraestructura de direcciones y enrutamiento eficaz y jerárquica, configuración de direcciones con y sin estado, seguridad integrada, mejora de la compatibilidad para la calidad de servicio, nuevo protocolo para la interacción de nodos vecinos, capacidad de ampliación.[25]

En la captura que se muestra a continuación se puede observar que a través del protocolo ICMPv6 se hace la solicitud de un vecino y se anuncia la presencia de otro equipo en la red Ad Hoc, el protocolo ICMPv6 es un protocolo que se encuentra en la capa de red.

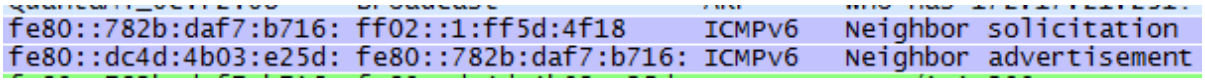


Fig. 28. Muestra un nuevo vecino en la red Ad Hoc que se ha configurado.

NBNS: NetBIOS Name Service, al utilizar la resolución de nombres de nodo punto a punto, todos los sistemas se registran con un servidor de nombres NetBIOS (NBNS), que es responsable de asignar el nombre del equipo a la dirección IP y se asegura que no existan registros de nombres duplicados en la red. Todos los sistemas deben conocer la dirección IP del NBNS, si los sistemas no están con la dirección IP correcta para el NBNS, la resolución de nombres de nodo punto a punto no funcionará. En caso de no tener acceso al NBNS no habrá forma de resolver los nombres y no se podrá tener acceso a otros sistemas de la red.[26]

A continuación se observa como el equipo con el nombre USUARIO-PC hace su registro con el servidor NBNS.

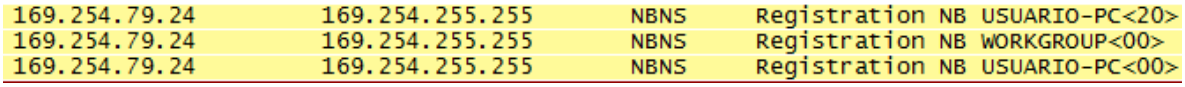


Fig. 29. Muestra el registro del equipo portátil USUARIO PC en el servidor NBNS.

TCP: Transmission Control Protocol, este protocolo se usa para manejar conexiones de extremo a extremo, garantizando que se entregarán los datos sin errores, en el mismo orden en que fueron transmitidos y con seguridad, es un protocolo de capa 4 del modelo OSI, se caracteriza por: Ser orientado a conexión, operación full-duplex, error checking, acknowledgements, flow control, y servicio de recuperación de paquetes.[27]

En la captura que se muestra a continuación se puede observar cómo se establece la conexión entre dos equipos, y se realiza transferencia de datos al aparecer activo el bit PSH, y finalmente se puede verificar como se da por terminada la conexión, cuando se transmiten el par de segmentos FIN y ACK.

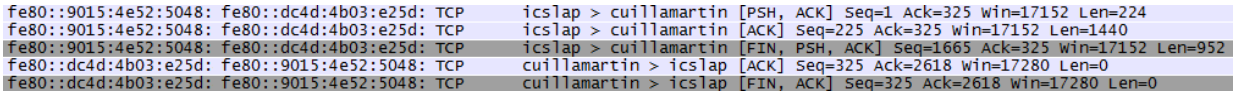


Fig. 30. Muestra como inicia y finaliza la conexión en la red Ad Hoc.

SSDP: Simple Service Discovery Protocol, es utilizado para descubrir en redes servicios basados en Universal Plug and Play (UPnP), que independientemente del fabricante, sistema operativo, lenguaje de programación, etc. permite el intercambio de paquetes a los dispositivos que están conectados a una red.

La arquitectura UPnP permite detectar automáticamente cualquier dispositivo que pueda ser incorporado a la red, obteniendo su dirección IP y su nombre, por otro lado le da a conocer a otros sus funciones y capacidad de procesamiento.[28]

A continuación se puede observar cómo se notifica en la red alguna de las condiciones antes mencionadas, se identifica una respuesta en la red como HTTP y al aparecer la instrucción “NOTIFY” se indica que se está anunciando en la red un nuevo servicio, si por el contrario la instrucción que aparece es “M-SEARCH” se indica que se está buscando un servicio en la red.

A continuación podemos observar los tipos de notificaciones que se generan a través del protocolo SSDP antes mencionadas.

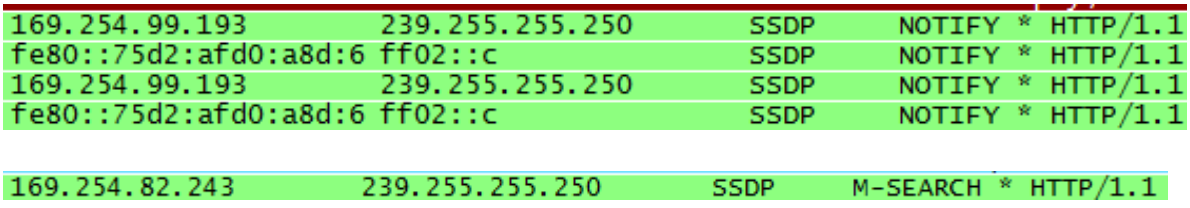


Fig. 31. Muestra las notificaciones generadas por el protocolo SSDP.

ARP: Address Resolution Protocol, es un protocolo de capa de enlace, se encarga de relacionar las direcciones físicas de los equipos (Ethernet MAC) con sus respectivas direcciones IP, para realizar esto envía un mensaje en broadcast a la dirección de difusión de la red que contiene la dirección IP por la que se pregunta, y se espera la respuesta de ese equipo o de otro con la

dirección Ethernet que le corresponde. Este protocolo crea tablas en donde cada interfaz está relacionada con una dirección IP y con una dirección física MAC.

Este protocolo se usa en cuatro casos específicos de comunicación: Cuando dos host se encuentran en la misma red y uno desee enviar un paquete al otro, cuando dos host están sobre redes diferentes y deben usar un gateway/router para comunicarse con el otro host, cuando un router necesita enviar un paquete a un host a través de otro router, y cuando un router necesita enviar un paquete a un host de la misma red. [29]

A continuación podemos observar como después de configurada la dirección IP en cada máquina virtual, el protocolo ARP inicia el proceso para descubrir las direcciones físicas MAC de estos, con lo que finalmente construirá las tablas ARP y tendrá la información completa de los equipos que se encuentran conectados a la red Ad hoc.

Fig. 32. Muestra los mensajes que usa ARP para relacionar en la red una dirección MAC con una dirección IP.

Capturas en la red Ad Hoc una vez se pone a correr el protocolo de enrutamiento AODV.

Una vez se pone a correr el protocolo AODV en la red Ad Hoc se generan las siguientes capturas.

A continuación podemos observar que el protocolo AODV que se encuentra corriendo en el equipo identificado con la dirección IP 192.168.1.6 inicia el proceso de descubrimiento de vecinos (equipos o nodos que se encuentren cerca) y rutas, transmitiendo en la red un mensaje route reply en modo broadcast (se indica que el destino del mensaje es 255.255.255.255). Este mismo proceso lo realizan los equipos que se encuentren en la red.

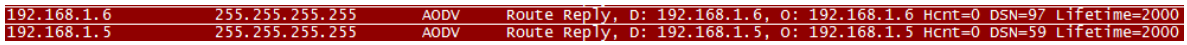


Fig. 33. Muestra la captura realizada en wireshark de los mensajes RouteReply que transmite el protocolo AODV.

En la siguiente captura se puede observar que cuando el protocolo AODV realizó la búsqueda de equipos (nodos) vecinos o que se encontraran cerca a él, no encontró ninguno generándose un mensaje route error tipo broadcast (para que los equipos en la red se informen del error que se genero) con la descripción unreachable destinations o destinos inalcanzables.

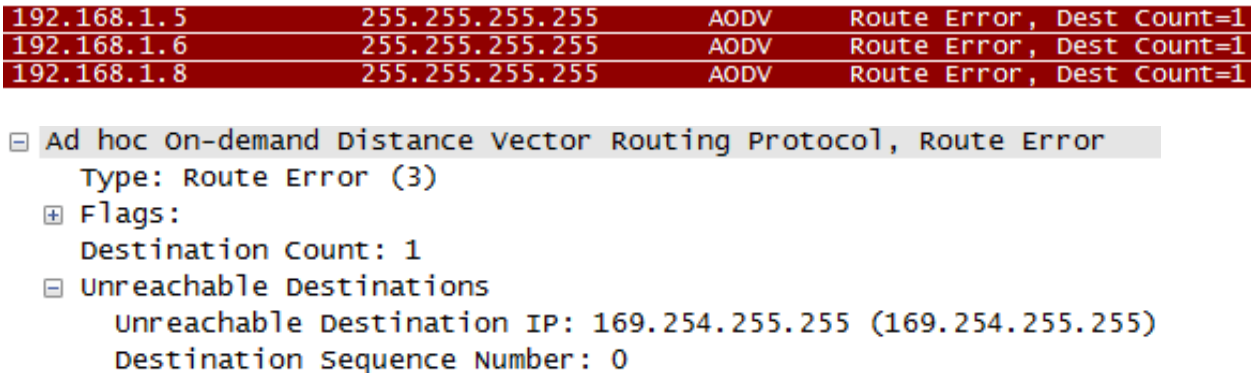


Fig. 34. Muestra la captura realizada en wireshark de los mensajes Route Error que transmite el protocolo AODV, y el detalle de dicha captura.

En la siguiente captura se puede observar que aunque el equipo con la dirección IP 192.168.1.6 recibe la ruta hasta el equipo (nodo) con la dirección IP 192.168.1.4, a través de un mensaje route request, que contiene información sobre los saltos que se deben dar hasta el destino (2 saltos), el número de secuencia y el número de identificación del route request (RREQ Id), dicha ruta presenta muy poco tiempo de vida (Time to live: 3), solo se aceptan rutas con Time to live:5 o mayor, por lo que este mensaje indica que la ruta que se ha encontrado no es confiable, este mensaje se envía en la red en broadcast, con el fin de que todos los nodos de la red sean informados.

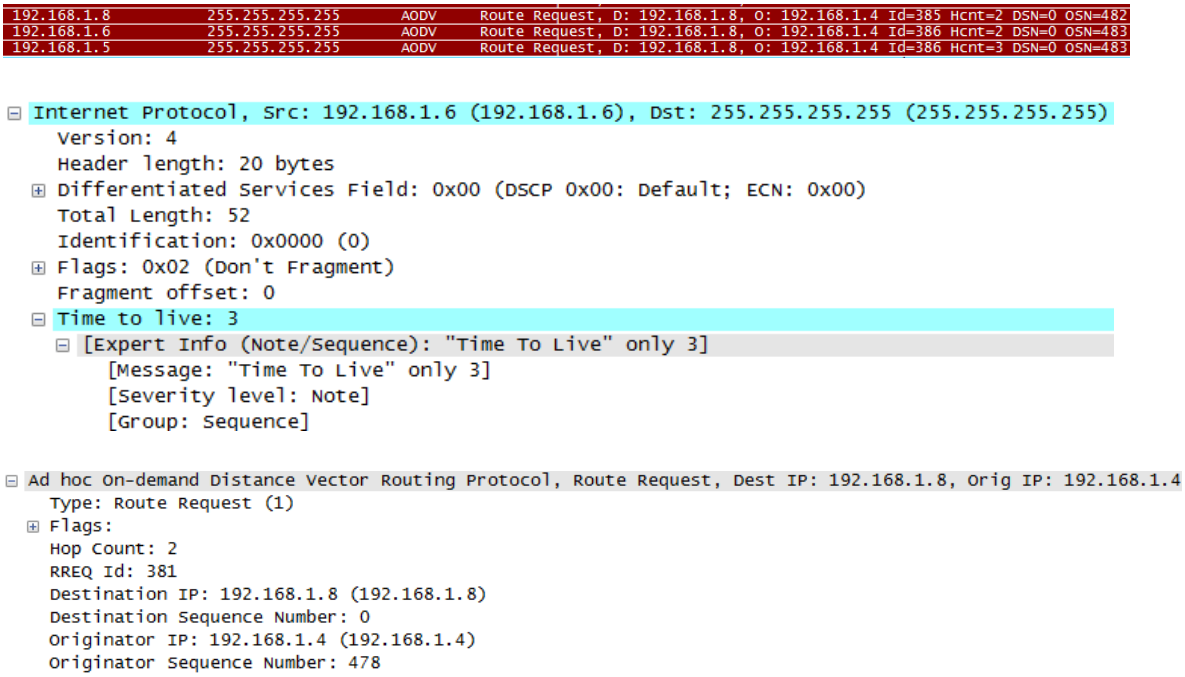


Fig. 35. Muestra la captura realizada en wireshark de los mensajes Route Request (erróneo) que transmite el protocolo AODV, y el detalle de dicha captura.

A continuación vamos a nombrar los equipos (nodos) para que el análisis de los mensajes route request, que transmite el protocolo AODV con las rutas que se deben usar para el intercambio de paquetes, sean más sencillos de entender, Los nombres de los equipos se encuentran en la Figura 26, a continuación podemos verificarlos.

A: 192.168.1.4	C: 192.168.1.6
B: 192.168.1.5	D: 192.168.1.8

Tabla 2. Muestra los nombres asignados a cada uno de los equipos a los que les corresponde la dirección IP que se muestra.

Las siguientes capturas fueron tomadas desde el equipo B que estaba tratando de establecer conexión con el equipo A. En esta captura se puede observar que los equipos dentro del área de cobertura de B recibieron una respuesta con las rutas hacia A, estas rutas se transmiten en la red en broadcast con el fin de que todos los equipos que conforman la red sean informados.

192.168.1.8	255.255.255.255	AODV	Route Request, D: 192.168.1.8, O: 192.168.1.4 Id=387 Hcnt=1 DSN=0 OSN=484
192.168.1.6	255.255.255.255	AODV	Route Request, D: 192.168.1.8, O: 192.168.1.4 Id=387 Hcnt=1 DSN=0 OSN=484
192.168.1.5	255.255.255.255	AODV	Route Request, D: 192.168.1.8, O: 192.168.1.4 Id=387 Hcnt=2 DSN=0 OSN=484

Fig. 36. Muestra la captura realizada en wireshark de los mensajes Route Request que transmite el protocolo AODV.

A continuación vamos a analizar cada una de las tramas capturadas: El primer equipo en encontrar una ruta hacia el equipo A, es el equipo D que tiene como vecino al equipo A, como se puede observar en el campo hop count esta a un salto de distancia. También se puede verificar esta ruta como confiable ya que el time to live del paquete que hallo la ruta no presento ningún error.

192.168.1.8	255.255.255.255	AODV	Route Request, D: 192.168.1.8, O: 192.168.1.4 Id=402 Hcnt=2 DSN=0 OSN=503

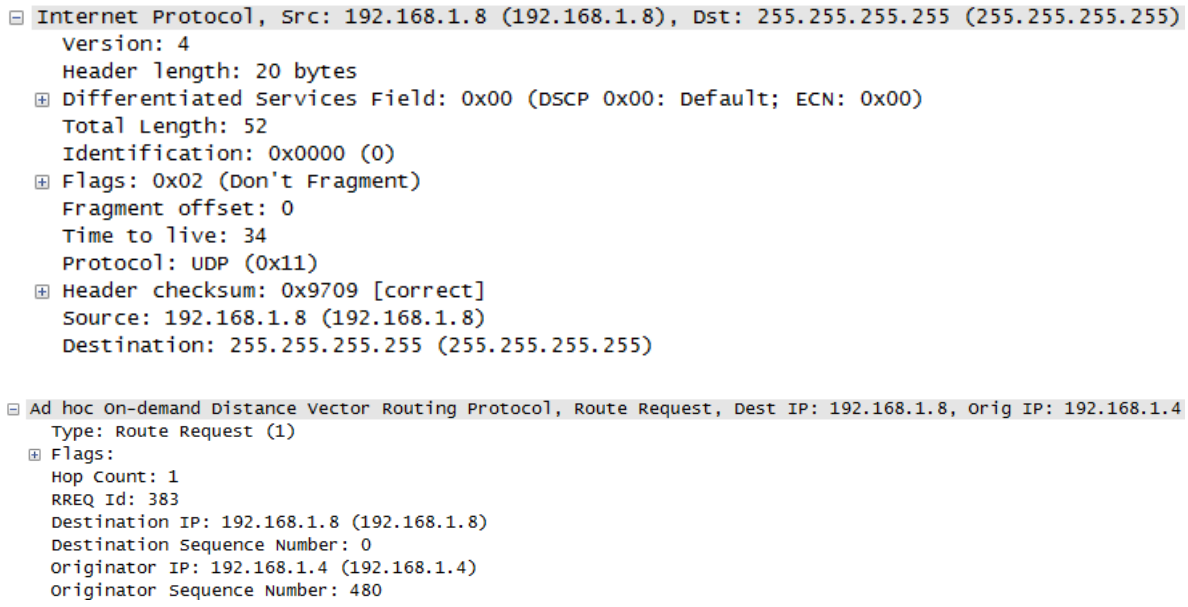
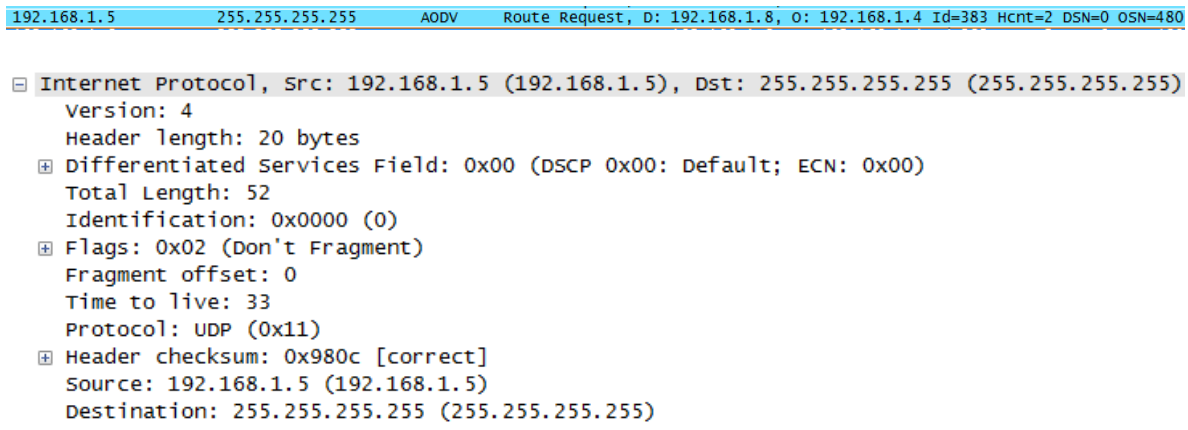


Fig. 37. Muestra la captura realizada en wireshark de los mensajes Route Request que transmite el protocolo AODV desde el equipo con dirección IP 192.168.1.8, y el detalle de dicha captura.

El siguiente equipo en transmitir a la red la ruta hacia A fue el equipo B (el equipo Origen), esta ruta la halló gracias a que su vecino el nodo D, había encontrado una ruta al nodo A (nodo destino), como se puede observar en la captura desde el nodo B el protocolo hace una cuenta de dos saltos hasta el nodo destino, adicionalmente se puede confiar en la ruta ya que el tiempo de vida de la ruta no presentó fallas.



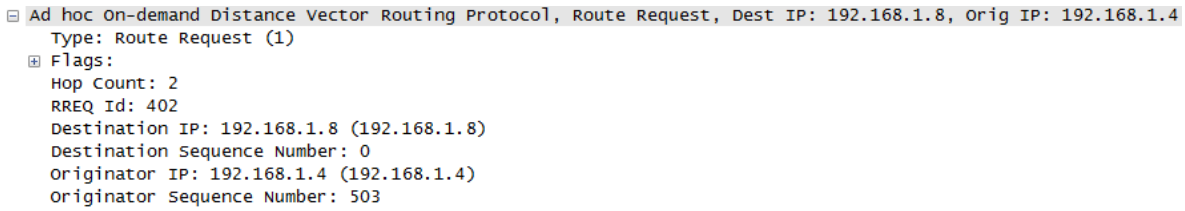


Fig. 38. Muestra la captura realizada en wireshark de los mensajes Route Request que transmite el protocolo AODV desde el equipo con dirección IP 192.168.1.5, y el detalle de dicha captura.

Finalmente, el último equipo en hallar una ruta hacia el destino fue el equipo C, esta ruta presenta un salto adicional a la anterior, aún así, es una posible opción que el protocolo encuentra. En la captura se puede observar que la ruta desde el nodo C presenta dos saltos hasta el nodo A, lo que indica que según la topología desde el nodo origen se contarían tres saltos hasta el destino.

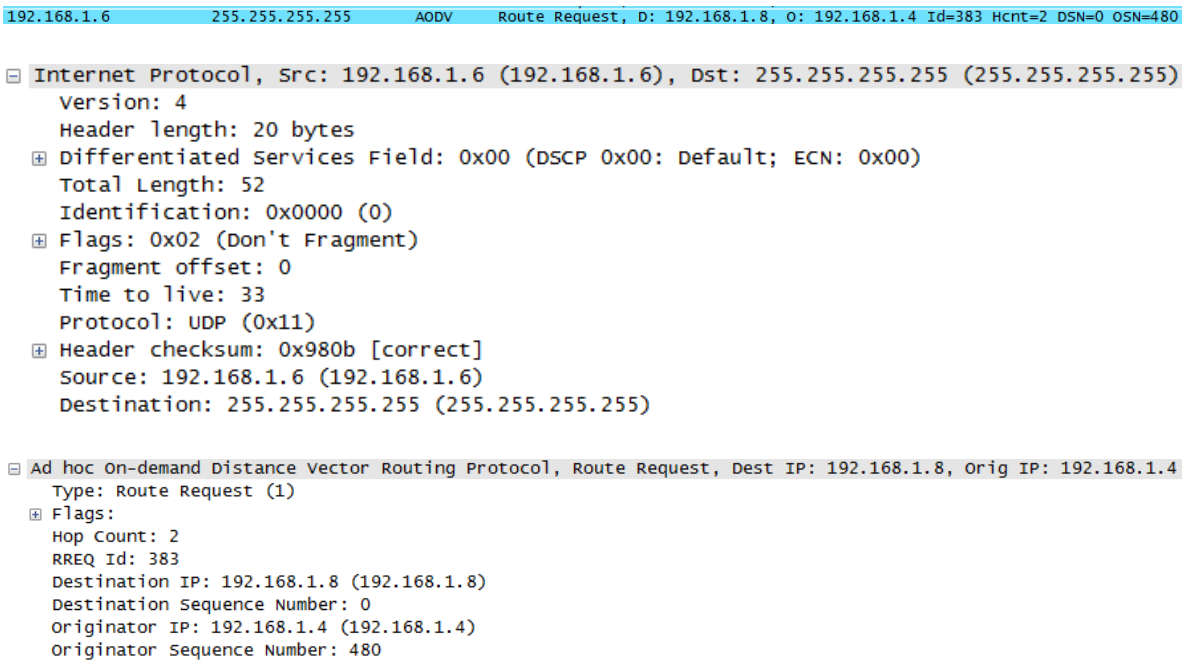


Fig. 39. Muestra la captura realizada en wireshark de los mensajes Route Request que transmite el protocolo AODV desde el equipo con dirección IP 192.168.1.6, y el detalle de dicha captura.

Finalmente, se debe mencionar que el protocolo de enrutamiento a pesar de realizar el proceso de descubrimiento de rutas, presentó fallas para la transmisión de paquetes y los equipos en la red sólo se pudieron comunicar con sus vecinos y no con equipos con 2 o más saltos de distancia, como se muestra a continuación, esto se debe a posibles fallas en el software que se implementó

del protocolo, esencialmente en la capa de transporte donde se debe realizar la retransmisión de los paquetes:

```
PING 192.168.1.5 (192.168.1.5) 1536(1564) bytes of data.
From 192.168.1.8 icmp_seq=15 Destination Host Unreachable
```

Fig. 40. Muestra que no se pudo alcanzar el equipo al que se deseaba hacer ping.

En la figura 40 se puede observar que el equipo con la dirección IP 192.168.1.8 le trató de hacer ping al equipo con dirección IP 192.168.1.5, que se encontraba a 2 saltos de distancia de este, lamentablemente no pudo realizar dicho ping porque el nodo era un “destino inalcanzable”, pero ya se verificó con anterioridad que el protocolo si encontró la ruta hasta este equipo deseado.

```
50 packets transmitted, 41 received, 18% packet loss, time 1217ms
rtt min/avg/max/mdev = 12.285/266.077/531.334/188.157 ms, pipe 24
```

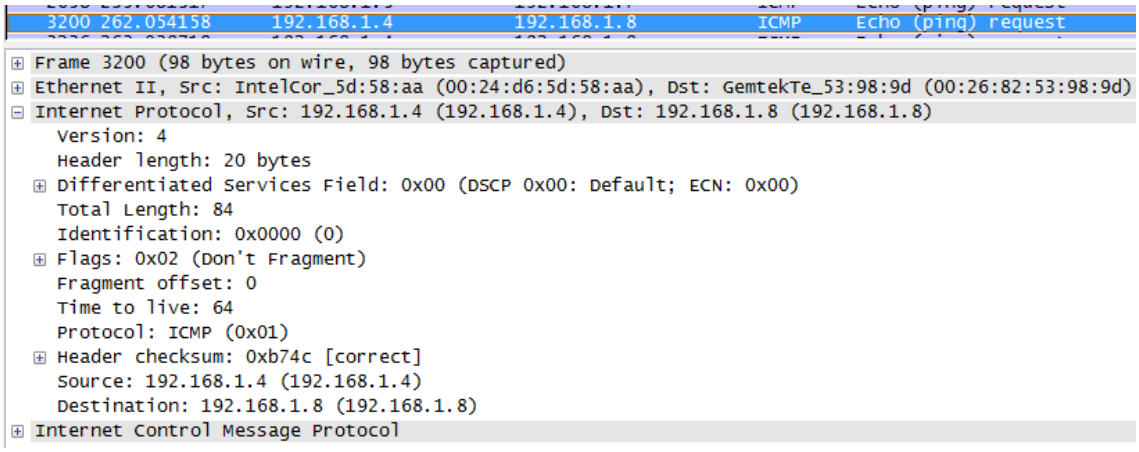


Fig. 41. Muestra el ping hecho a un nodo vecino.

En la figura 41 se puede observar que efectivamente el equipo con dirección IP 192.168.1.4 si le pudo realizar ping al equipo vecino con dirección IP 192.168.1.8, aunque se presentaron perdida de paquetes, se pudo capturar en la red Ad Hoc el ping a través de wireshark.

A continuación se muestran las estadísticas generadas una vez se transmitieron paquetes hacia un vecino:

PING	p tx	p rx	s (Byte)	%pp	%pe	rtt	rtt min	promedio rtt	rtt max	de rtt
# ping –s 512 –c 15 –i 0.01 192.168.1.5	15	15	512	0%	100%	179	392,423	2758,708	3848,035	1471,053
# ping –s 1536 –c 15 –i 0.01 192.168.1.5	15	1	1536	93%	7%	385	1866,951	1866,951	1866,951	0
# ping –s 2048 –c 15 –i 0.01 192.168.1.5	15	7	2048	53%	47%	306	853,944	1660,249	2463,581	554,058
# ping –s 512 –c 50 –i 0.01 192.168.1.5	50	41	512	18%	82%	1217	12,285	266,077	531,334	188,157
# ping –s 1536 –c 50 –i 0.01 192.168.1.5	50	50	1536	0%	100%	1919	15,382	164,058	389,433	118,514
# ping –s 2048 –c 50 –i 0.01 192.168.1.5	50	30	2048	40%	60%	1499	14,944	88,994	169,833	42,179
# ping –s 512 –c 15 –i 1 192.168.1.5	15	13	512	13%	87%	14134	8,171	73,452	376,749	98,376
# ping –s 1536 –c 15 –i 1 192.168.1.5	15	15	1536	0%	100%	14112	9,483	119,107	711,976	170,512
# ping –s 2048 –c 15 –i 1 192.168.1.5	15	15	2048	0%	100%	14104	20,886	146,329	683,615	167,988
# ping –s 512 –c 50 –i 1 192.168.1.5	50	48	512	4%	96%	49430	10,181	121,359	646,386	140,157
# ping –s 1536 –c 50 –i 1 192.168.1.5	50	13	1536	74%	26%	49547	35,804	1323,12	5822,289	1982,15
# ping –s 2048 –c 50 –i 1 192.168.1.5	50	48	2048	4%	96%	49315	15,95	321,224	3064,438	523,587

Tabla 2. Muestra las estadísticas generadas al transmitir los paquetes ping a un nodo vecino.

Convenciones	
s	Tamaño de cada paquete en Byte
c	Cantidad de paquetes
i	Frecuencia con que se envian los paquetes en segundos
rtt	tiempo ida y vuelta en ms
p	paquete
de	desviación estandar
pp	paquete perdidos
pe	paquete entregados

Tabla 3. Muestra las convenciones usadas en la tabla de estadísticas y en los paquetes ping que se transmiten.

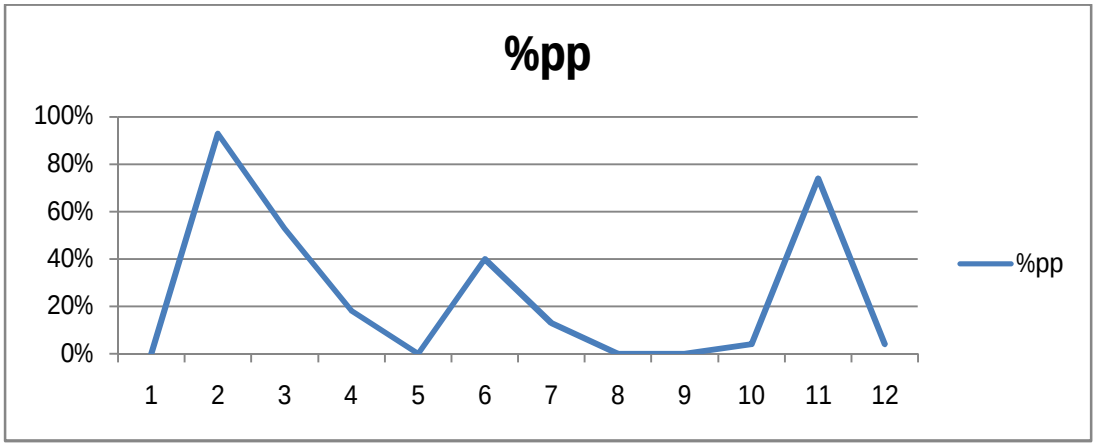


Fig. 42. Muestra el porcentaje de paquetes perdidos al transmitir los paquetes ping a un nodo vecino.

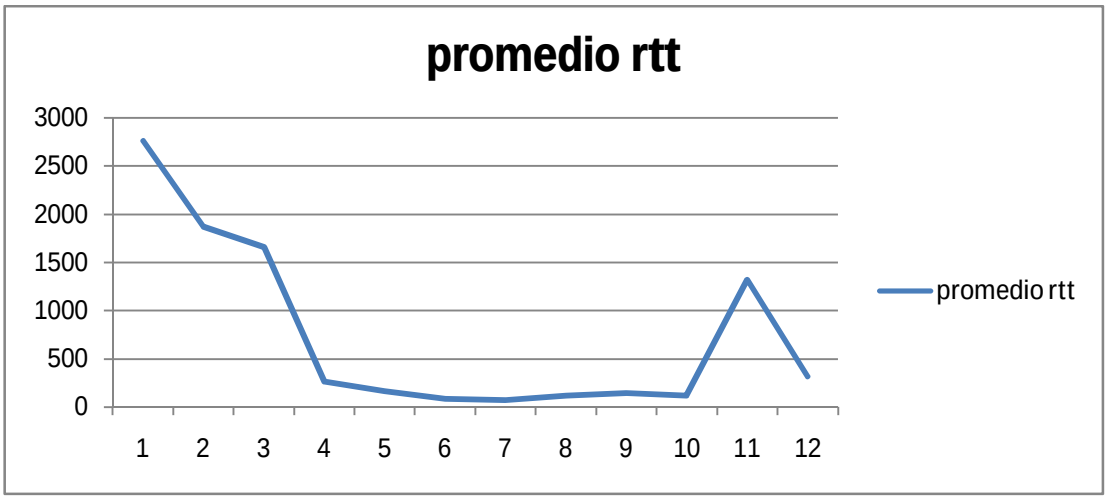


Fig. 43. Muestra el retardo promedio en que se transmitieron los paquetes y se confirmo en el nodo origen que fueron recibidos.

En las figuras 42 y 43 se puede observar que los paquetes perdidos no muestran un patrón de comportamiento, por el contrario hubo paquetes que fueron entregados casi en un 100% y otros que no se entregaron, de igual manera se pueden observar grandes variaciones en el retardo promedio de vuelo de los paquetes, por lo anterior se puede analizar la inestabilidad del protocolo de enrutamiento AODV.

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

Se concluyó que uno de los factores más influyentes para el buen funcionamiento del protocolo de enrutamiento es el rango de cobertura de cada uno de los nodos que conforman la red, ya que si los nodos tienen muy poco rango de cobertura se desconectan de la red fácilmente y obligan al protocolo a reiniciar la búsqueda de nuevas rutas con frecuencia, las pruebas que se realizaron mostraron que en espacios cerrados con gran flujo de gente y obstáculos el rango de cobertura de los nodos en línea recta fue hasta de 20m y tratando de sobre pasar paredes de 10m sin que los equipos se desconectaran de la red.

Gracias a la implementación del protocolo AODV se pudo definir que presenta gran inestabilidad, las rutas no siempre se hallan de forma adecuada pues las interferencias del ambiente hacen que los mensajes de descubrimiento de rutas se tarden en llegar a su destino, el protocolo toma esto como una error y reacciona a esto desechando las rutas e iniciando nuevamente el descubrimiento de rutas, esto se puede observar en la figura 35 en donde el nodo destino intentaba enviar su ruta al nodo origen, a través de un route request, pero por los motivos anteriores no podía completar el time to live adecuado para proceder a la transmisión de datos.

El protocolo AODV-UU inunda la red con mensajes HELLO, que se encargan de verificar si las rutas ya establecidas aun se encuentran activas, estos mensajes los debe activar y desactivar el usuario manualmente, lo que genera retardo y posibles fallas en la transmisión de datos, para que el protocolo presente un mejor funcionamiento los mensajes HELLO deben activarse automáticamente cuando se inicie la transmisión de datos.

El protocolo se probó en máquinas virtuales con y sin entorno gráfico, concluyéndose que las diferentes aplicaciones que se cargan en las máquinas con entorno gráfico hacen que el nodo se desconecte de la red Ad Hoc con gran frecuencia, como es el caso de los software que al detectar que la señal de la red a la que se encuentra conectado el equipo es muy baja, auto buscan y conectan el equipo a la red que tenga mayor señal, y en ciertos momentos simplemente se desconecta el equipo de la red Ad Hoc, y en consecuencia el protocolo no corre de la manera adecuada.

Como se mostró anteriormente, el protocolo de enrutamiento hace correctamente el descubrimiento de rutas, mostrando en las capturas la cantidad de saltos que debería hacer el paquete para llegar a su destino, pero existen fallas que deben ser corregidas en la capa de transporte, ya que los mensajes ICMP no son transferidos al destino a través de nodos intermedios, pero si se transmiten a nodos vecinos con algunos paquetes perdidos.

EL protocolo AODV-UU presenta problemas para guardar la configuración de las direcciones IP que se le asignan a los nodos de la red, estas direcciones en ocasiones son reemplazadas por la dirección 0.0.0.0 y generan mensajes en broadcast con rutas que no corresponden a ningún nodo, este error se puede observar en los mensajes que el protocolo envía o en un analizador de protocolos.

El buen funcionamiento del protocolo AODV se ve afectado por los obstáculos que se encuentren en el ambiente, en este caso disminuyó su rendimiento al correr en un entorno cerrado y con gran flujo de personas, además, se debe tener en cuenta que en el entorno que se probó (Universidad Militar Nueva Granada) existe gran cantidad de equipos móviles que añaden interferencia a la red.

Para ambos protocolos los equipos deben estar configurados en la misma red con direcciones IPv4, de lo contrario no existirá manera de establecer conexión entre ellos y mucho menos de enrutar paquetes en la red, esto se debe a que aún los protocolos de enrutamiento se encuentran en desarrollo y aún no aceptan direcciones IPv6 para establecer conexión con otros equipos.

Para el protocolo DSR se debe aclarar que se corrigieron errores de compatibilidad, ya que la versión del protocolo no ha sido actualizada desde el año 2005, y aún corrigiendo los errores mencionados no se pudo establecer comunicación con otros equipos cuando se encontraban conectados a una red Ad Hoc, ya que esta versión no es aún compatible con estas redes y aunque no se presentan errores los mensajes no son transmitidos ni recibidos, por lo que se prosiguió a implementar el protocolo en un sistema operativo más reciente, siendo imposible corregir la gran cantidad de errores que se generan y presentándose el mismo problemas sobre una red Ad Hoc, es importante aclarar que el protocolo se desarrolló para ser implementado en un sistema operativo específico y al usar otra versión para su implementación este no se adapta.

El sistema operativo en el que se puede montar el protocolo DSR, al ser obsoleto no se pudo instalar como partición física en un equipo portátil, ya que se presentaban otro tipo de errores que hubieran entorpecido la labor de probar el protocolo, por lo que finalmente este protocolo no se pudo probar en equipos con partición física.

Es de gran importancia resolver los problemas antes mencionados, como la inestabilidad de los protocolos, fallas en la capa de transporte, el rango de cobertura de los nodos, entre otros y realizar más pruebas que verifiquen el real funcionamiento de los protocolos de enrutamiento, además, de actualizar los protocolos, para que se pueda establecer conexión a través de direcciones IPv6 y en redes mucho más grandes, y que sean menos dependientes de la versión del sistema operativo que se esté usando. Gracias al estudio realizado se podrá desarrollar nuestro propio protocolo de enrutamiento en java, que además será compatible con android.

1. BIBLIOGRAFIA

[1] Barakovic, Sabina and Barakovic, Jasmina (2010), “Comparative performance evaluation of Mobile Ad Hoc routing protocols”, *MIPRO, 2010 Proceedings of the 33rd International Convention.*

[2] Jingfang, Su and Muqing, Wu and Jingrong, Wen (2009), “Comprehensive Evaluation of AODV in Wireless Multi-Hop Ad Hoc Networks”, *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09.5th International Conference on.*

[3] Kulkarni, S.A. and Rao, G.R. (2008), “Mobility Model Perspectives for Scalability and Routing Protocol Performances in Wireless Ad-Hoc Network”, *First International Conference on Emerging Trends in Engineering and Technology.*

[4] León, Miguel Saumett and Castro Barrera, Harold Enrique (2007), “Análisis de desempeño del protocolo de enrutamiento DSR bajo diferentes modelos de movilidad”, *Épsilon enero-junio 008.*

[5] Mahmood, A.S.M.A. and Hossain, M.S. and Aziz, F.A. (2008), “Comparative performance analysis of DSR and AODV protocol based on mobility factor”, *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conferenceon.*

[6] Medina Santos, A (2006), “Comparativa de los protocolos AODV y OLSR con un emulador de redes ad-hoc”.

[7] Mittal, S. and Kaur, P (2009), “Performance Comparison of AODV, DSR and ZRP Routing Protocols in MANET'S”, *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies.*

[8] Murazzo, M.A. and Rodríguez, N.R. and Martínez, M (2010), “Evaluación del retardo de los protocolos de ruteo reactivos para redes Manet”, *Revista de Ingeniería Electrónica, Automática y Comunicaciones.*

[9] QianFeng and ZhongminCai and Jin Yang and Xunchao Hu (2009), “A Performance Comparison of the Ad Hoc Network Protocols”, *Computer Science and Engineering, WCSE '09, Second International Workshop on.*

[10] Rajput, M. and Khatri, P. and Shastri, A. and Solanki, K (2010), “Comparison of Ad-hoc reactive routing protocols using OPNET modeler”, *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on.*

[11] Runcai Huang and YiwenZhuang and Qiying Cao (2009), “Simulation and Analysis of Protocols in Ad Hoc Network”, *Electronic Computer Technology, 2009 International Conference on.*

[12] Shah, N. and Qian, D. and Iqbal, K. (2008), “Performance evaluation of multiple routing protocols using multiple mobility models for mobile ad hoc networks”, *Multitopic Conference, 2008. INMIC 2008. IEEE International.*

[13] Ugas, Chalmeta Jordi (2009), “Estudio y análisis de prestaciones de redes móviles Ad Hoc mediante simulaciones NS-2 para validar modelos analíticos”, UNIVERSIDAD POLITECNICA DE CATALUÑA.

[14] Villalba, G. and Javier, L (2007), “Estudio del rendimiento y la seguridad en redes ad hoc”.

[15] Xu, Fengying and Zhimin, Liu and Yongchun, Xu (2010), “A Comparative Study on Mobile Ad Hoc Wireless network routing protocols simulation”, *volume 978-1-4244-7941-2/10/$26.00 ©2010 IEEE.*

[16] Yeo, In-Ho and Kim, Yong-Won and Rhee, Jong Myung and Kim, H.-A. and Yang, Hyo-Sik (2009), “Performance analysis of routing protocols in Mobile Ad-hoc Networks under group mobility environment”, *Advanced Communication Technology, 2009. ICACT 2009.11th International Conference on.*

[17] WANG Lin-zhu, FANG Ya-qin, SHAN Min (2009), “Performance Comparison of Two Routing Protocols for Ad Hoc Networks”, *IEEE.*

[18] Mercado, Armando and Berríos Figueroa, Rafaelgil and Chan Ye, Paul, “Redes inalámbricas ad hoc”, Universidad Interamericana de Puerto Rico.

[19] Frodigh, Magnus and Johansson, Per and Larsson, Peter (2000), “Formación de redes inalámbricas ad hoc—El arte de la formación de redes sin red”, *Ericsson Review No. 4.*

[20] Bartolomé Arquillo, Diego (2006), “ANÁLISIS DE LA DISTRIBUCIÓN DE TRÁFICO EN REDES MÓVILES AD HOC CONECTADAS A INTERNET”, Universidad de Málaga escuela técnica superior de ingeniería de telecomunicaciones.

[21] Villacrés Torres, (2009), “Análisis del desempeño de una red WPAN Basado en el estándar IEEE 802.15.4 utilizando Network Simulator 2”, ESCUELA POLITÉCNICA DEL EJÉRCITO, DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA, CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES, PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA, SANGOLQUÍ – ECUADOR.

[22] Perkins, Charles E. y Belding-Royer, Elizabeth M. y Das, Samir R. (2003), “AD HOC ON-DEMAND DISTANCE VECTOR (AODV) ROUTING”, Universidad de California y Universidad de Cincinnati, RFC 3561 experimental.

[23] Johnson, David B. y Maltz, David A. y Hu, Yih-Chun, (2007), “THE DYNAMIC SOURCE ROUTING PROTOCOL (DSR) FOR MOBILE AD HOC NETWORK FOR IPV4”, RFC 4728 experimental

[24] Aboba, B. y Thaler, D. y Esibov, L. (2007), “Link-Local Multicast Name Resolution (LLMNR)”, RFC 4795 Informativo.

[25] Conta, Alex y Deering, Stephen y Gupta, Mukesh (2006), “Internet Control Message Protocol (ICMPv6)”, RFC 4443 Normas Track.

[26] Network Working Group (1987), RFC 1001.

[27] Information Sciences Institute y University Of Southern California y 4676 Admiralty Way y Marina del Rey, California 90291 (19981), “Protocolo de Control de Transmisión”, RFC 793.

[28] Cai, Ting y Leach, Paul y Gu, Ye y Goland, Yaron Y. (1999), IETF INTERNET DRAFT.

[29] Plummer, David C. (1982), “An Ethernet Address Resolution Protocol”, RFC 826.